2

PROCESS DEFINITION AND MODELING GUIDEBOOK

VOLUME 1

CONCEPTS AND PRINCIPLES OF MPDM

SPC-92041-CMC

VERSION 02.00.02

MARCH 1994

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE March 1994	3. REPORT TYPE AND DATES COVERED Technical Report

4. TITLE AND SUBTITLE Process Definition and Modeling Guidebook	5. FUNDING NUMBERS G MDA972-92-J-1018
6. AUTHOR(S) R. Bechtold, J. Brackett, S. Redwine Produced by Software Productivity Consortium under contract to Virginia Center of Excellence	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES) Virginia Center of Excellence SPC Building 2214 Rock Hill Road Herndon, VA 22070	8. PERFORMING ORGANIZATION REPORT NUMBER SPC-92041-CMC, Version 02.00.00
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) ARPA/SISTO Suite 400 801 N. Randolph Street Arlington, VA 22203	10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES None	
12a. DISTRIBUTION / AVAILABILITY STATEMENT No Restrictions	12b. DISTRIBUTION CODE 1

13. ABSTRACT (*Maximum 200 words*)

This two volume guidebook provides guidance on efficiently developing and evolving a quality set of process definitions that will direct and improve the ways you develop software. This guidebook is particularly helpful for those persons who must fulfill a software process improvement action plan calling for definition of part or all of their software process.

This guidebook provides a flexible set of templates and techniques for capturing and representing processes and explains how to use the template-based information to produce guidebooks training materials and models. Basic usage principles and techniques are presented in Volume 1; advanced usage concepts are presented in Volume 2.

14. SUBJECT TERMS Process, software engineering			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

PROCESS DEFINITION AND MODELING GUIDEBOOK

VOLUME 1

CONCEPTS AND PRINCIPLES OF MPDM

SPC-92041-CMC

VERSION 02.00.02

MARCH 1994

Produced by the
SOFTWARE PRODUCTIVITY CONSORTIUM SERVICES CORPORATION
under contract to the
VIRGINIA CENTER OF EXCELLENCE
FOR SOFTWARE REUSE AND TECHNOLOGY TRANSFER

SPC Building
2214 Rock Hill Road
Herndon, Virginia 22070

CONTENTS

FIGURES

TABLES

EXAMPLES

This page intentionally left blank.

PREFACE

The technology described in this guidebook is part of a broad approach to software productivity improvement. This preface provides an overview of that approach and identifies the series of guidebooks that support it. These guidebooks were developed by the Software Productivity Consortium under contract to the Virginia Center of Excellence for Software Reuse and Technology Transfer (VCOE). For a complete listing of VCOE guidebooks and products, call the Software Productivity Consortium's Technology Transfer Clearinghouse at (703) 742-7211.

Each technology has been packaged so it can be used without reference to the other technologies. However, it is also possible to combine these technologies into an integrated approach for product development. Figure P-1 shows how the guidebooks for these technologies relate to the practices of software development organizations.

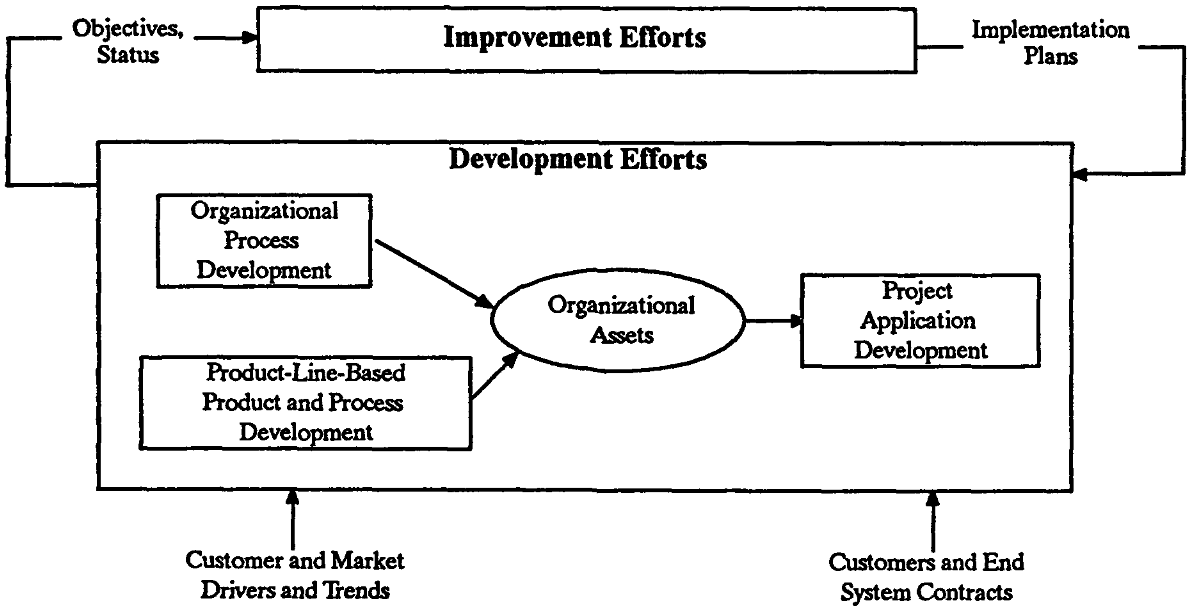

Objectives, Status
Improvement Efforts
Implementation Plans
Development Efforts
Organizational Process Development
Product-Line-Based Product and Process Development
Organizational Assets
Project Application Development
Customer and Market Drivers and Trends
Customers and End System Contracts

Figure P-1. Structure for Integrated Application of Consortium Technologies

These practices are composed of:

- *Improvement Efforts (IE).* Application of technology to improve software development efforts. These efforts require managed approaches to assessment of objectives and current capabilities, planning for the improvement, implementation of the plan, and measurement of success.
- *Development Efforts.* Development of products that meet the needs of customers and markets or products that make the organization more competitive in meeting expected future needs.

- *Organizational Process Development (OPD).* Development of standardized organizational process assets (e.g., process and method descriptions, process enactment tools) tailored for a particular organization.

- *Product-Line-Based Product and Process Development (PLD).* Development of integrated product and process assets (e.g., core products and processes for adapting them for particular customer needs) appropriate for a particular product line.

- *Project Application Development (PAD).* The tailoring and application of organizational assets for a particular product development effort.

Table P-1 describes how existing products can be integrated to address your organizational practice.

Table P-1. Consortium Guidebooks and Related Practices

Guidebook	**Part Number**	**Relationship to Software Practice**
Consortium Requirements Engineering Guidebook	SPC-92060-CMC	Used for defining and analyzing requirements in PAD. Adaptable for use in PLD.
Managing Process Improvement: A Guidebook for Implementing Change	SPC-93105-CMC	Supports IE by providing a process and supporting guidance ꞏor nitiating and maintaining an organizational process improvement program.
Process Definition and Modeling Guidebook	SPC-92041-CMC	Provides methods for defining and documenting processes so they can be analyzed, modified, and enacted. Supports IE and OPD.
Process Engineering With the Evolutionary Spiral Process Model	SPC-93098-CMC	Used to iteratively plan, manage, and control PAD and PLD. Used to construct organization-specific processes in PLD and tailor them in PAD.
Reuse Adoption Guidebook	SPC-92051-CMC	Supports IE by providing specific process improvement activities for incorporating reuse practices.
Reuse-Driven Software Processes Guidebook	SPC-92019-CMC	Provides development approaches for PLD (domain engineering) and PAD (application engineering) of reusable software assets.
Software Measurement Guidebook	SPC-91060-CMC	Supports IE by providing methods for quantitative assessment of project status.
Using New Technologies: A Technology Transfer Guidebook	SPC-92046-CMC	Supports IE by providing a process that addresses how to get an organization to use new technologies.

This guidebook considerably expands upon the first version to reflect lessons and examples from use, and to increase coverage in a number of areas. For example, the number of templates has more than doubled, the number of data fields has increased nearly tenfold, and the definition of process risks and process metrics has been given explicit support. Additionally, template usage has been separated into four tiers of formality, thereby allowing you to easily select and use only those templates and fields that directly support your process definition goals.

ACKNOWLEDGMENTS

Volume 1 of this guidebook was authored by Richard Bechtold and John Brackett. Volun ıs authored by Richard Bechtold, John Brackett, and Sam Redwine.

The Consortium would like to thank the Vitro Software Engineering Process Group members: Walt Greenspon, Rick Emery, Pauline Martin, Herb Nachmann, and Ed Zarrella and the Vitro Corporation for their participation in a process definition pilot project. Valuable insights were gained—and numerous improvements made—in the Consortium's process definition methodology and techniques as a direct result of their work.

The Consortium would also like to thank Alice Forey and Jennifer McLaughlin, SYSCON SEPG members. Their early use of the revised process templates likewise resulted in significant improvements to the material presented in this guidebook.

Thanks go to Consortium personnel Mary Eward, Donna Garfield, and Chuck Lynch for their extensive reviews and recommendations for improving this guidebook.

Special thanks go to Bobbie Troy, Consortium Technical Editor, whose efforts substantially improved the organization and presentation of this information. Thanks also go to Debbie Morgan and Deborah Tipeni for incorporating the markups to the draft document and to Betty Leach and Tina Medina for clean proofing the final copy.

This page intentionally left blank.

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

The objective of this guidebook is to guide you in efficiently developing and evolving a quality set of process definitions that will direct and improve the ways you develop software—both to improve your products and process and to allow assessment at the DoD Software Engineering Institute's (SEI) Capability Maturity Model (CMM) Levels 2 (Repeatable) and 3 (Defined). You should be better able to engineer your process descriptions and the processes themselves.

1.1.1 A Practical Approach to Process Definition

This guidebook offers a practical approach to reflecting experience and distilled research, and includes highlights from a significant, real-world example. This approach has several characteristics that help ensure a successful process definition and modeling effort. For example, Managed Process Definition Methodology (MPDM):

- Provides "how to" guidance for early success and a sound foundation for continuing success
- Organizes the definition as it is developed
- Avoids unnecessarily elaborate or formal notations
- Is driven by your goals
- Considers your context and the factors you need to deal with in your situation
- Exploits simple automation to rapidly generate and revise guidebooks, training material, etc.
- Directly supports management (including self-management) of the process definition effort

Organizations need to have a representation option that combines the strengths of text- and graphic-based representations while minimizing their respective weaknesses. MPDM is specifically designed to support process representation as three separate yet tightly integrated concerns: (1) design, development, and implementation of the process model; (2) generation of various output products, such as guidebooks and training material, from your process model; and (3) management and review of the development of both the process model and the derived output products.

The data-organizing templates presented in this guidebook can be rendered as paper templates, but their design directly facilitates "automated templates" via mainstream information management repositories. Examples of template fields include text-oriented descriptions of activities, pre- and post-conditions, internal processing, comments, role descriptions, product descriptions, risk types and levels, revision history, etc. However, through a variety of relationships the templates also convey an explicit architecture directly supporting graphical rendering and analysis. This combined template- and graphically-based approach provides you greater opportunity for the optimal combination of both text and diagrams toward the cost-effective development and use of process representations.

1.1.2 Process Definition Goals

There are numerous goals for process definition:

- To improve your processes organization-wide such as through a Total Quality Management (TQM) initiative
- To reduce software management or technical problems
- To respond to customer or market pressures to improve or certify such as SEI CMM or ISO 9000
- To reach aggressive business goals involving software
- To augment or replace existing process documentation that is either unused or too expensive to maintain

Process definitions are needed for the same reasons sports teams need playbooks. A team without a playbook must transfer everything from head to head typically through long apprenticeship, is only as good as what is in their heads, has more trouble adjusting and improving plays, scratches their changes only in the dirt, and can never be on the same page. Though they do different things with them, playbooks are important to both coaches and players—an essential organizational asset.

Every software organization, regardless of size or maturity, has a process for developing and maintaining software. When using a defined software process, your organization may experience some of the following benefits:

- ***Improved productivity (and teamwork)*** because communication among the process users, managers, process developers, and customers is more effective
- ***Reduced rework*** because you identify and eliminate problems early in the process rather than later
- ***Efficient project staff start-up time*** because there is a documented process to train them on
- ***Reduced software development costs*** due to reduced volatility in software development processes
- ***Improved predictability of budgets and schedules*** because you have defined what to measure, when to measure it, and how to use the information
- ***Improved tool usage*** because tool usage is defined and supported by training material
- ***Faster project start-up*** because the project has a process that it can tailor

- ***Increased integration between resulting products*** due to improved coordination among and communication between teams
- ***Increased quality of the resulting products*** because reviews are defined and understood to be an integral part of the process

Existing examples of process representations include policy, procedures, and operational manuals developed by organizations to inform and guide their employees in the performance of their responsibilities. Most organizational process guidebooks only define the process, but a few make use of relatively high-level or simple process models.

Although process modeling is a comparatively rare technique for representing organizational processes, it is a well-known and mature technique for representing systems implemented by computer systems. Example techniques include Statecharts, Structured Analysis and Design Technique (SADTs), and the Entry-Task-Validation-eXit (ETVX) paradigm. Due to fundamental parallels between defining and modeling organizational processes and computer-based systems processes, you can apply many techniques from systems process representation to organizational process representation. Similarly, many of the advantages and benefits derived by building computer process representations can also be derived from organizational process representations.

1.1.3 Goal-Driven Approach

Unfortunately, probably as many attempts at process definition fail as succeed. The job is not simple or intuitive and can fail in a number of ways. These ways to fail include: undertaking the wrong scope; trying to define too much too soon; ignoring existing staff, process, or culture; over or under designing of process; using inappropriate techniques for process definition; developing an inconsistent or rigid definition; producing guidebooks that are not used; updating too slowly; and loss of sponsorship or support due to cost or schedule overruns within the process definition effort.

To avoid these types of failures, you need to clearly define specific goals. Example goals include:

- Analyze process (in)efficiencies
- Identify and remove process redundancies
- Identify and eliminate areas where the process is unknown or undefined
- Gain insights into process risk
- Identify where, when, and how process metrics will be collected and used

The key point is that it is important to allow your goals to drive your process definition effort. If the goal is to analyze the representation for process bottlenecks, this will influence the type of representation that should be constructed. If the goal is to identify and remove process redundancy, to eliminate areas where the process remains largely undefined, seek insights into process risk, or to establish or analyze a metrics program, all these considerations influence not only the type of definition or model constructed but also the approach taken in researching and constructing those representations. This issue is examined in detail in Section 5, Volume 2.

Process representation can span the effort spectrum from easy to exceedingly difficult as a function of process complexity and desired level of detail:

- Initially, you may find that relatively simple diagrams of isolated parts of your process are sufficient to achieve your immediate goals.
- Later, you may find that you can achieve additional goals by extending your existing representation, adding more detail, and maybe capturing primary interrelationships between your growing inventory of "process asset" models.
- Still later, other goals may be achieved through even further expansion of scope, detail, abstraction, and information.

The magnitude of your goals must be directly related to the magnitude of organizational support for process representation. This guidebook, and the tools and techniques it proposes, have been designed to facilitate precisely this type of goal-driven, incremental approach to process definition and modeling.

1.2 INTENDED AUDIENCE AND USES

The intended audience for this guidebook includes practitioners interested in developing process definition(s) that are acceptable for SEI Level 2 or 3 assessments, ISO-9000 certification, and those interested in the tangible benefit derived from applying process definition and modeling techniques and methods including automation. This audience is primarily:

- Software Engineering Process Group (SEPG) members
- Process Action Teams
- Process engineers and modelers

Line engineers, project managers, and anyone working on or interested in the area of process analysis, design, development, improvement, or management can also use this guidebook (see Table 1-1 a mapping of topics to audience). Because most of the intended readers are not necessarily familiar with the issues and aspects of process definition, representation, and modeling, motivation and rationale behind the recommended techniques are provided throughout the guidebook.

To clarify what sections you should read, Table 1-1 identifies generic audience types and the sections of this guidebook that each should read. Keep in mind that at any one time you may fit the description of more than one audience type.

Table 1-1. Guidebook Audience

	Volume 1			Volume 2						
Audience Type	1	2	3 – 6	1	2	3	4	5	A	B
Person doing basic process definition	✓	✓	✓	✓					✓	✓
Person leading basic definition	✓	✓	✓	✓	✓					
Person doing advanced definition	✓	✓	✓	✓		✓	✓	✓		
Person leading advanced definition	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Person performing process modeling			✓				✓			✓
Person controlling resources	✓	✓		✓	✓	✓				

1.2.1 Scope of Use

This guidebook can help you:

- Establish a common foundation for process engineering, training, and documentation
- Develop process-oriented guidebooks, improve their usability, or reduce their cost
- Develop process training and education
- Improve your process either in general, for ISO 9000 certification, or to CMM Level 2 or 3
- Construct process representations either generally or based upon SADT, ETVX, or another process representation paradigm
- Model your processes

While this guidebook can be useful in a variety of contexts, it particularly benefits people who must develop a software process improvement action plan calling for definition of parts or all of their software processes.

This guidebook introduces both fundamental and advanced process definition concepts, explains how to develop a definition, and provides nontechnical as well as technical advice. While you can use this guidebook by itself, you are also provided with relevant references to other material.

1.2.2 Levels of Use

This guidebook provides a flexible set of templates and techniques for capturing and representing processes and explains how to use the template-based information to produce guidebooks, training materials, and models. The material is presented on two primary levels:

- Basic usage: suitable for you if you need to produce process guidebooks and training material
- Advanced usage: suitable for those having increased need of process formality (to support, for instance, process simulation and automated enactment)

Volume 1 presents basic usage principles and techniques; Volume 2 presents advanced usage concepts. Volume 1 conveys the entire MPDM. Volume 2 provides more details and, in some areas, extends the scope. Although Volume 2 addresses issues related to automated process enactment, the principle subject of this guidebook is how to prepare process definitions that can be effectively used by people.

1.3 GUIDEBOOK ORGANIZATION

This guidebook is separated into two volumes. Volume 1 covers the basics suitable for organizations needing to produce process definitions intended to be enactable by humans as opposed to machines. Volume 1 contains Sections 1 through 6.

The structure of Volume 1 is:

- Section 1, Introduction, describes the goals, scope, and audience of this guidebook.
- Section 2, Process Definition Basics, presents the role of process definition within an organization and fundamental process definition concepts; it shows where and how you use process definitions and introduces MPDM.
- Section 3, MPDM Templates and Graphical Notation, describes the conceptual model behind the information gathering templates, goal-based process definition, and template "tiers of usage." Then it discusses in detail the Tier 1 templates and their associated fields.
- Section 4, Fundamentals of Defining Your Process, explains how to start using the templates and describes in detail the eight tasks of MPDM.
- Section 5, Examples: Peer Review Process, presents a complete set of examples for MPDM-based process definition and modeling. These examples include a context diagram, indented lists, graphical models, completed templates, a navigation and extract algorithm, and an example of automatically generated pages to a guidebook.
- Section 6, Automated Process Definition and Modeling, discusses the use of mainstream, commercial off-the-shelf tools that support MPDM and briefly discusses a process definition pilot project.
- The List of Abbreviations and Acronyms contains abbreviations and acronyms used in this guidebook and their definitions.
- The Glossary contains a list of terms used in this guidebook and their definitions.
- The References section contains sources cited in this guidebook.
- The Bibliography section contains additional sources of information.

Volume 2 covers a more advanced approach suitable for those with prior process definition experience, focuses on process definition issues from the larger perspective of general process improvement, and contains information on defining processes that are enactable by machines. Volume 2 contains Sections 1 through 5 and two appendixes.

The structure of Volume 2 is:

- Section 1, Introduction, describes the goals, scope, and audience of this guidebook.
- Section 2, The Road to a Defined Process: The Management Perspective, discusses organizational change, the introduction of process definition and modeling into an organization, the scope of organizational process definition, and briefly discusses metrics.
- Section 3, Organizational Process Definition, presents the goals of organizational process definition; a top-down approach to process definition; process engineering; process architectures; and an alternative, bottom-up, "pockets of excellence" approach to process definition.
- Section 4, Advanced Templates, presents the full set of process definition templates and describes Tier 2, Tier 3, and Tier 4 usage.

- Section 5, Alternative Process Representations, describes characteristics of process representations, degrees of formality, choosing the "right" process notation, alternative notations and models, and the benefits of MPDM.
- Appendix A, Inspection Templates Example, provides a completed set of templates to support the example in Section 5 of Volume 1.
- Appendix B, Checklists, provides a set of checklists that you can tailor to facilitate your process definition effort.
- The List of Abbreviations and Acronyms contains abbreviations and acronyms used in this guidebook and their definitions.
- The Glossary contains a list of terms used in this guidebook and their definitions.
- The References section contains sources cited in this guidebook.
- The Bibliography section contains additional sources of information.

1.4 AUTHOR INTERACTION AND FEEDBACK

The Consortium is actively interested in end-user reaction, case studies, feedback, pilot projects, and any suggestions or recommendations you have for improving and advancing the content of this guidebook. While insights from researchers and technologists are also appreciated, the Consortium is especially interested in feedback from those producing process definitions. The Consortium encourages you to apply the information presented, and strongly encourages feedback on how to make this information progressively more useful to you and others. Contact Software Productivity Consortium, SPC Building, 2214 Rock Hill Road, Herndon, Virginia 22070 (703) 742-7211. Ask for (or write to) the Process Definition Technology Manager in the Process Improvement Division.

1.5 TYPOGRAPHIC CONVENTIONS

This guidebook uses the following typographic conventions:

Serif font	General presentation of information.
Initial Capitalization, Serif font	Names of fields, values within fields, activities, and work products.
CAPITALIZED SERIF font	Names of undesired events.
Italicized serif font	Mathematical expressions and publication titles.
Boldfaced serif font	Section headings and emphasis.
Boldfaced italicized serif font	Run-in headings in bulleted lists and, in the Appendix, minor subsections.
Sans serif font	Information ordering.

This page intentionally left blank.

2. PROCESS DEFINITION BASICS

2.1 THE ROLE OF PROCESS DEFINITION

Defining your software process is crucial to the success of your process improvement effort. This section covers the context, concepts, and activities of process definition and prepares you for the remainder of the guidebook. Generally, process definition and modeling activities are conducted in the context of process improvement and process engineering.

2.1.1 PROCESS DEFINITION AS PART OF PROCESS IMPROVEMENT

Defining your process is typically part of a process improvement effort. Many process improvement plans call for incrementally defining a process as an early step in an improvement program and a requirement to meet assessment or certification goals. Success of process definitions depends on their fit to, and acceptance by, all the affected persons and organizational units as well as their technical merit. Only by understanding and involving all the stakeholders, combined with process competence, can success be expected. Recognized need to change, management support, a strong influential advocate, adequate resources, and supportive initial users are all important to obtain maximum value from a process definition effort.

Process definitions have many uses, including guiding persons to do successful work and providing an explicit representation of a process that can be analyzed and changed in the pursuit of improvement. While it may be difficult to achieve, a definition of an improved process is still only a limited part of improving the process. The process of defining a process also impacts process improvement.

To be consistently successful, process definition efforts need to be part of a well-planned and implemented process improvement effort. *Managing Process Improvement: A Guidebook for Implementing Change* gives extensive guidance on establishing a well-managed process improvement effort. It provides a cyclic process for process improvement involving 15 major activities in 5 groups, see Figure 2-1. The guidebook shows how you can develop a well-structured and executable plan by considering a set of core activities you can plan and execute in an iterative manner. Figure 2-1 presents these steps:

- ***Understand Context.*** Understand the current context, including support for improvement, and the current development process. A process assessment is frequently performed as part of understanding the current context.
- ***Analyze Risks and Select Strategy.*** Analyze risks in continuing to follow the current process. Select a strategy for incremental implementation of improvements to the present process.
- ***Plan Improvements.*** Create a prioritized action plan for improvement and commit the resources required to implement the action plan.

- ***Implement Improvements.*** Implement the process improvements according to the action plan and monitor projects in order to monitor the impact of the process improvements.
- ***Review and Update.*** Review progress and decide how to proceed with the next process improvement iteration.

Multiple improvement iterations are required since process change must be gradual and continuous. Because the culture within a organization must change, evolution from current practice is preferable to massive process changes.

The process definition activities described in this guidebook fit mainly into the Understand Context and Implement Improvements activities shown in Figure 2-1. In the Understand Context activity, the present process must be understood to provide the foundations for process improvement decisions. In the Implement Improvements activity, an improved process will be defined as part of carrying out an action plan.

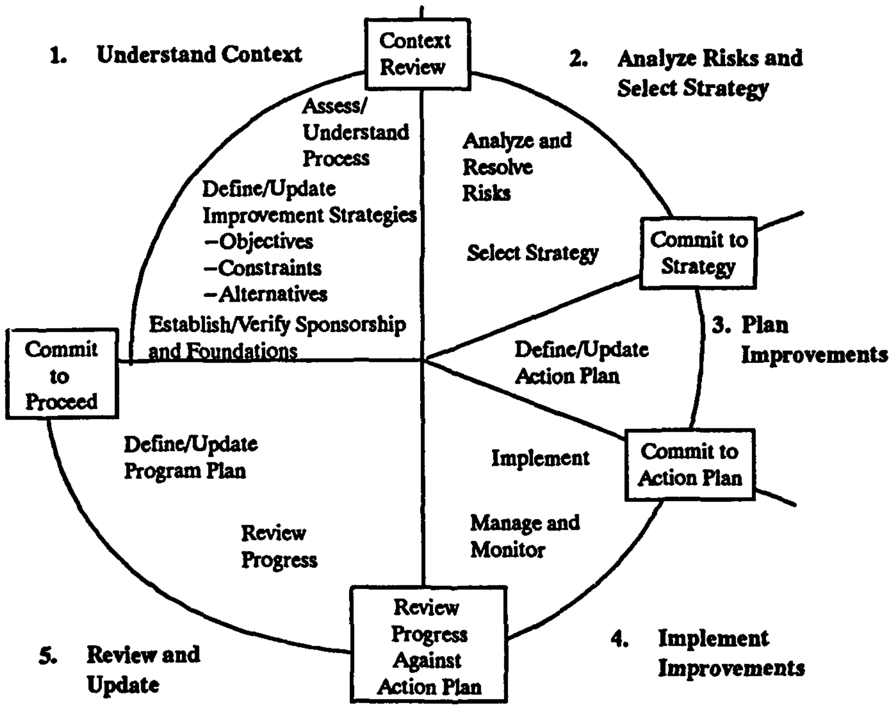

1. Understand Context
Context Review
2. Analyze Risks and Select Strategy
Assess/ Understand Process
Analyze and Resolve Risks
Define/Update Improvement Strategies
–Objectives
–Constraints
–Alternatives
Select Strategy
Commit to Strategy
Establish/Verify Sponsorship and Foundations
3. Plan Improvements
Commit to Proceed
Define/Update Action Plan
Define/Update Program Plan
Implement
Commit to Action Plan
Review Progress
Manage and Monitor
Review Progress Against Action Plan
5. Review and Update
4. Implement Improvements

Figure 2-1. Process Improvement Process

2.1.2 Process Definition as Part of Process Engineering

Software is developed and evolved using a "system" involving complex processes. Experience has shown that for consistent quality results such complex processes need to be engineered using organizationally sensitive approaches, knowledge of the domain and relevant technology, openness to change, and thorough understanding of users and requirements. This guidebook is, in part, a contribution to the relatively young field of software process engineering, but much of the advice provided is similar to what one would also find in successful systems and industrial engineering.

You should build on what you already may know about how to do systems analysis, design, verification and validation, and the introduction of improved technology in similar organizational settings. Examples you can learn from include:

- The process of discovering what is really happening in an organization
- The advantages of iterative over big-bang approaches
- The importance of risk management
- The essential role of configuration management
- The need to provide for variations for use in different situations
- The centrality of people and organizational issues to success

Thus, process definition is part of process engineering, which is described in detail in *Process Engineering With the Evolutionary Spiral Process Model* (Software Productivity Consortium 1994).

The Process Definition and Modeling Guidebook describes the subset of the activities in process engineering that support process definition. To better understand what is to be done in such a project, and this guidebook's relationship to the activities to be performed, you need to understand a number of basic concepts related to process definition.

2.2 CONCEPTS

Process definitions exist at different levels of generality within different levels of an organization, can be expressed using a variety of representations, and are developed for different purposes.

2.2.1 Levels of Process Definition

As shown in Figure 2-2, process definitions can exist at several levels, including organizational, business area, program, and project levels. Process improvement activities can start at any level. In many cases process definitions are first prepared for a subset of the activities performed on a project. For example, a detailed definition of an improved configuration management process might be developed on a project and then generalized at the program level to be applicable to projects of different sizes. Alternatively, the configuration management process might be defined at the program level and provided to projects with guidance on how to specialize or tailor it based on project size and implementation environment.

Over time, an organization will build up "process assets" which assist projects in creating a project-specific process. Using the terminology of the SEI, these assets can be divided into three groups:

- *Software Process Kernels.* Kernels specify fundamental activities which are likely to be performed on many projects (i.e., configuration management).
- *Software Life-Cycle Models.* These models define the sequence of activities performed during the software life cycles appropriate for an organization's software products. For example, one project may elect to use the "waterfall" model, while another may decide to implement an incremental development life-cycle model.

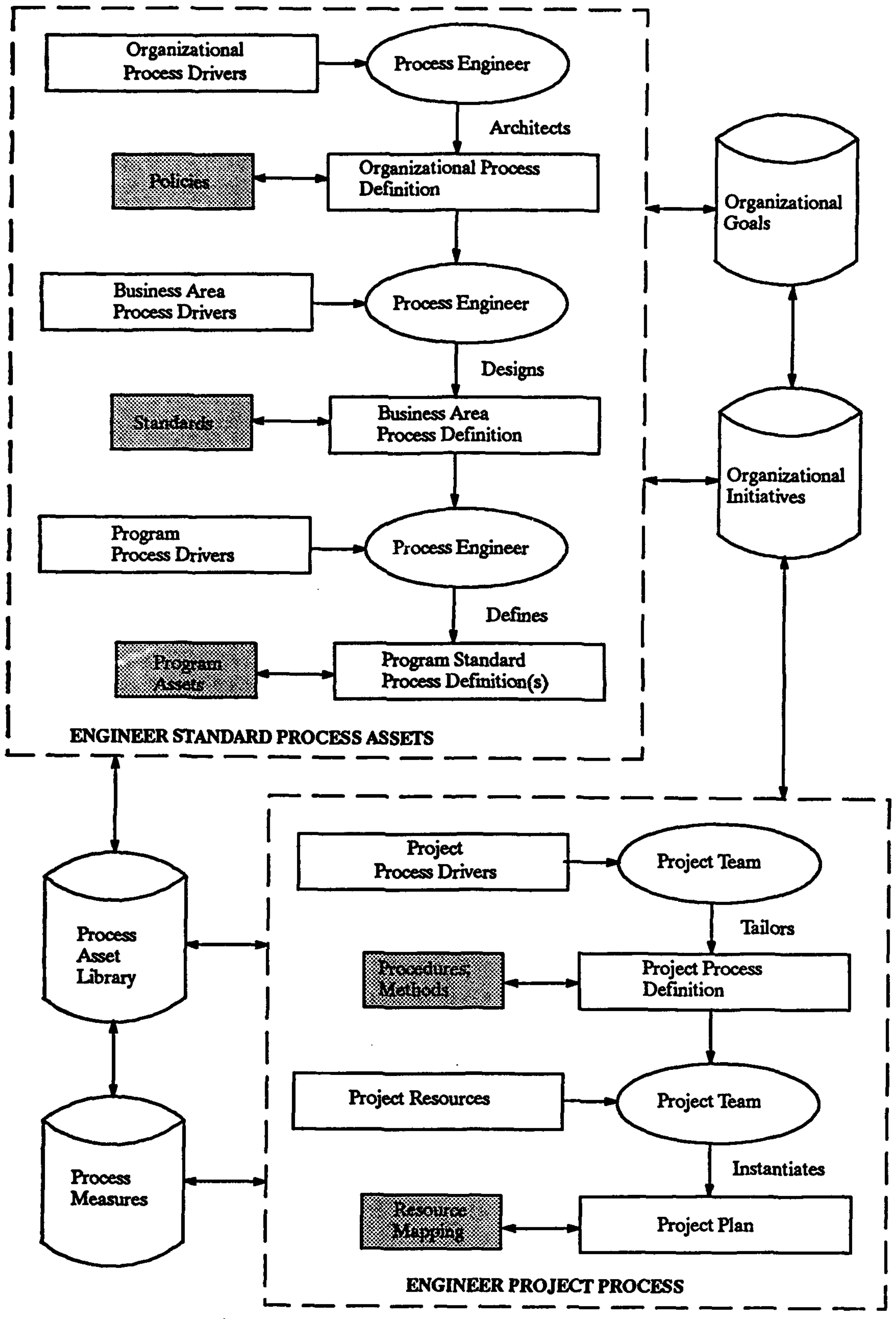


Figure 2-2. Abstraction Levels for Process Definitions

- *Process Architectures*. At one or more levels, process architectures show major process steps or relationships, how those steps or relationships interact, and how process kernels fit within the architecture.

The guidance and techniques presented in this guidebook apply to all three types of assets.

2.2.2 Contents of Process Definitions

Section 3 describes the contents of process definitions in detail. However, any process definition must describe what the process does, who does it, what products and resources are needed, and what products are produced. Detailed process definitions should contain sufficient information for the process to be successfully carried out (i.e., enacted) as defined. The definition will usually require, or assume the existence of, a certain level of personnel skills and appropriate resources. Skills and resources essential for successful enactment need to be described in the process definition.

At a minimum, this guidebook describes a process in terms of three useful abstractions:

- "Activities" which are performed
- "Products" to indicate the things used and produced
- "Roles" to describe who performs the activity

A process definition will describe many relationships among these three types of abstractions; some examples are:

- One activity verifies the work of another activity.
- A role performs an activity.
- A product is produced by an activity.

Note: Products produced as a result of process definition and modeling are referred to as end-products.

Processes are dynamic in nature and, therefore, describing process behavior over time is a critical part of developing a process definition. You need to include the following types of process behavioral information in your process definition:

- Under what circumstances can an activity begin
- What must happen for a product to be considered "approved"
- What are the entry and exit criteria for an activity
- Where can you choose from a set of alternative activities

An activity description describes both **what** is to be done and **how** it is to be accomplished. The steps for accomplishing the activity can be represented in many ways ranging from a textual description to detailed algorithmic specification. Alternative descriptions of "how" may be provided; for example, the activity "requirements definition" could be carried out for a particular application by using an object-oriented analysis method. In many cases, a process definition will reference separately documented procedures or methods.

2.2.3 Process Definition End-Product Formality

A process definition document (or a process guidebook) is written for the people who will enact the process. Process definitions can also be written for execution by computers, and are frequently called process programs. However, the primary subject of this guidebook is how to prepare process definitions that can be effectively used by people.

The staff defining the process (i.e., process engineers) must gather a large amount of information and organize it in order to produce understandable guidebooks. Various modeling techniques have been developed which help the process engineer with the information gathering and organizational tasks. For a modeling technique to be useful to process engineers it must describe:

- The relevant abstractions: activities, products, and roles
- The required relationships among these three abstractions of the process
- The behavior of the activities, products, and roles over time

Process definition end-products can be characterized by their degree of formality:

- Unconstrained descriptive text
- Template-based descriptions
- Constrained graphical models
- Executable graphical models or process grammars
- Mathematical notations

The most common technique for representing processes is the use of text. However, the potential for inconsistency, ambiguity, and uncertainty is considerable. Therefore, an increasing number of process engineers are using more precisely defined notations to support their work in understanding and analyzing processes.

Having graphical support for a notation can directly contribute to having some degree of formality. Usually, a graphically oriented notation comes coupled with a method that imposes rules on placement, use, and connections between the graphical objects. These rules constrain the types of structures that can be built and increase the formality of the resulting depictions. Additionally, graphical depictions are especially useful for portraying abstract or high-level relationships. The most frequently used graphical techniques used by process engineers are IDEF0, SADT, and Structured Analysis. The techniques described in this guidebook are compatible with and supportive of generating diagrams consistent with these (and other) techniques. However, in the event you want to avoid the complexities involved in generating such formalized pictures, there is a less formal graphical notation presented in Section 3. Finally, extremely formal descriptions involve extensive use of mathematical notations.

An important capability of a modeling method is the ability to represent how processes can be broken down, or decomposed, into subprocesses. Process decomposition usually is done by decomposing the activities and identifying the relevant products and required resources with each subactivity. One of the principal reasons for using a graphical modeling method is to show several levels of decomposition and the relationships between the subprocesses (i.e., how the output of one activity is used as an input to another activity).

Most modeling techniques use formal or semiformal notations that are too obscure or complex for practitioners to accept in guidebooks. A guidebook should describe what its intended reader needs to know and should be written so that the reader can easily access needed information. Therefore, the preparation of a usable guidebook involves much more work than just repackaging a process definition model prepared by a process engineer. Sections 3 through 6 provide suggestions on developing and maintaining usable guidebooks.

2.2.4 Goal-Driven Process Representation

As introduced in Section 1.1.3, process representation spans the entire spectrum from very easy to exceedingly difficult as a function of process complexity, intended usage, and desired level of detail. Before an organization undertakes process definition and modeling, it needs to clearly define the audience for the process definition results and the specific objectives that it expects. If the goal is process reengineering to improve productivity and reduce process redundancy, the type of definition or model constructed by process engineers is likely to be significantly different than if the goal is to produce training materials for a process that remains largely undefined. Accordingly, the approach taken in researching and constructing these representations will vary.

The magnitude of your goals should be directly related to the magnitude of organizational support for process representation. If the organization can only nominally support a process representation effort, the goals you strive for must be kept realistically small. Conversely, if an organization is prepared to give considerable support, then the goals of your process representation effort can be correspondingly larger.

The key point is that it is important to allow your goals to drive your process representation effort. Initially, you may find that relatively simple diagrams of parts of your process are sufficient to achieve your immediate goals. Later, you may find that you can achieve additional goals by extending your existing process representations, adding more detail, and perhaps packaging the representation as a process kernel that can be tailored for use on multiple projects.

Understanding the audience for your work will help define your goals. A process definition for use by staff with experience in performing the process can focus on providing detail in high risk areas, while providing minimum constraints in areas where good professional judgment is an adequate guide. If your goal is to produce a process guidebook and training materials for new, junior employees, more detail in most process steps will be required. The MPDM presented in this guidebook has been designed to facilitate this type of goal-driven, incremental approach to process definition and representation.

2.3 OVERVIEW OF PROCESS DEFINITION ACTIVITIES

MPDM is applicable across a wide spectrum of organizations with different process engineering needs. In a typical organization, some organizational standards based on a widely used life-cycle model may exist, but the standards may only specify milestones to be completed and the characteristics of products to be produced. Since few reusable, enactable software process kernels exist in most organizations, a potential starting point for process definition work is creating process kernels needed for one project which are likely to be useful to other projects. Example areas for creating such kernels are configuration management, testing, project planning, peer reviews, and quality assurance.

Unfortunately, the typical maturity of the process definition process itself is initially low. This guidebook emphasizes elements that help you start effectively and improve with experience. If you are aiming toward having a defined software process, then the process definition effort should define its process too. Section 4.2 describes the tasks associated with MPDM in detail. For now, however, we can consider process definition to consist of following five activities:

- Process End-Product Definition
- Process Definition Planning
- Process Familiarization
- Iterative Process Definition Development
- End-Product Generation

2.3.1 Process End-Product Definition

The objective of this activity is a clear definition of the end-product(s) that will result from the process definition effort, the goals of these specific end-products, their audience, scope, level of detail, etc. Achieving a clear, shared explicit statement of the goals that reflect the user's needs for the process definition product is a key to success. Setting goals that are compatible with the resources available to the process engineers is also important, and must be the basis for all process definition planning. One strategy is to initially generate the process definition end-product at the project level, test it with project staff, and then develop a tailorable version for use throughout an organization. Producing the organizational version will typically involve removing project-specific detail and focusing on the constraints that must guide any implementation of the process. As a consequence, process definitions at the organizational level should be sparse, "low-side compliant" process descriptions that preserve, and pass through to the project level, the greatest amount of freedom possible.

If, for example, the process end-products will be a configuration management guidebook and a training course, important issues to process engineers deciding the goals for these products would be:

- Whether the products are intended for both small and large projects
- Whether the audience is the entire development staff or just configuration management specialists
- What type of automated support will be assumed

Some typical questions you might ask about the audience are:

- Who will use the process definition end-product?

- What do they need to know to fulfill their current responsibilities?
- What vocabulary do they use?
- What experience do they have in performing the current process?
- How much effort are they willing to put into understanding the process definition end-product?

If the planned audience cannot enact the process efficiently and correctly, the process definition effort will fail. One of the goals of MPDM is to facilitate end-product verification by rapid, iterative generation of process definition end-products. This is achieved through automated support using low-cost, commercially available tools (e.g., database, word processing, and graphics packages). Since producing the "perfect" guidebook the first time is impossible, your goal should be to efficiently create increasingly effective versions as you obtain (pilot) project feedback.

2.3.2 Process Definition Planning

The objective of this activity is to verify that the scope of work is consistent with the resources available, and the necessary products can be produced within the necessary and time and budget constraints.

Establishing the scope of the process definition effort needs careful consideration, particularly at the beginning. Among the factors to consider are the objectives for the product, resources available, time period available for the effort, cost and benefits and cooperation expected from information sources outside the process definition team. You need to verify your initial statement of scope against process improvement plans and validate it with all affected parties. Although overall management support may exist for the process improvement effort, you should also seek sponsorship from the management of the portion of the organization that must assist the process definition team and who will use the process definition end-product.

The process definition effort should be treated as a "miniproject" with a project plan, measurable milestones, a schedule, and resource requirements. Agreement is needed on how to validate the completeness, correctness, and usability of the process definition products. You will select the method and notation to be used by process engineers. You will also decide how the process definition end-products will be produced by the process engineers.

2.3.3 Process Familiarization

The objective of this activity is to obtain as much context as possible for the process definition work. This activity consists of three major subactivities:

- Review existing process documentation
- Identify roles of key personnel in the process
- Develop process context diagram

With minimum impact on current operations, you can gather a great deal of information about the current process and the local terminology from existing process documentation, even if the existing documents are underutilized. Other useful sources are corporate standards, contractually binding regulations, organizational charts, and position descriptions. Your goal is to locate the most relevant materials, review them, and gain an understanding of the key activities and products involved in the process. You also want to identify the roles of key personnel and organizations in performing the process.

Some useful questions to ask to become familiar with the process are:

- What are the primary products and services provided by the process?
- What organizations are the consumers of these products and services?
- What organizations provide products, services, and resources needed to perform the process?
- What are all the organizational elements involved in performing the process?

The answers to these questions will help you assess the value of the information you have collected and to make a plan for gathering missing information. Interviews are helpful for gathering additional information whether you are documenting an existing process or defining an improved process for future adoption. Interviews will probably identify many hidden requirements and constraints that must be satisfied for a process to be readily accepted.

Also valuable is the use of review teams to carry out a review process that will build consensus regarding the actual or desired process. The ideal team should include process practitioners and senior technical and managerial staff with in-depth knowledge of the process. There are fields within the process templates specifically intended to facilitate this review process.

A useful way to organize the information you have been gathering is to create a diagram that describes the current flow of products between the process being defined and the external consumers and producers. Once you have confirmed these external interfaces to the process, make a list of three to ten major activities and the most important products involved in the current process. Later you may redefine these activities and products, but this initial overview of the process will help you organize the materials you need to gather. Section 5 presents an example of this approach, with further details.

2.3.4 Iterative Process Definition Development

The objective of this activity is to rapidly develop your process definition while maintaining a high level of end-user involvement and feedback.

This activity consists of three major subactivities:

- Data gathering and analysis of results
- Model construction
- Verification and review

The key to successful data gathering is to find out what is "really going on," not what is supposed to be going on. If you need to conduct interviews, a useful way of obtaining maximum value from the interview is to have the subject describe the steps of producing a product that is an external output of the process and to describe their role in performing the process activities. Ask the interviewee to think in terms what has to be accomplished before the activity can take place, what the process inputs and outputs are, and what the criteria are for completion of the activity.

In the second subactivity, model construction, process engineers prepare a representation of the process to understand and analyze the process and to communicate with the review team. MPDM, including the template-based process description described in Section 3, is one possible representation approach. Alternatives are IDEF0 and ETVX which are briefly described in Volume 2 of this guidebook.

The result of the model construction subactivity is not the process definition end-product. Whatever the form of model used, the last step in the process definition process will be to produce the process definition end-product to be used by process practitioners. Whatever approach you select, you must carefully consider how that approach supports end-product generation. Having elaborate IDEF0 diagrams is of questionable value if only 1% of your audience can—or is willing to take the time to—correctly interpret those diagrams.

Building a model helps the process engineer examine the relationships within the information gathered and identify errors, missing data, or inconsistent data. The evolving model will document the current state of the process engineer's understanding of the process and will focus attention on where more information gathering is required. The process of building a model is described in detail in Section 4.

The objective of the verification and review subactivity is to determine whether the process definition work faithfully represents the process being described and whether the model is a satisfactory basis for producing the process definition end-products. A recommended part of the verification process is a walkthrough of the model by the process engineers to show the reviewers what the model is intended to communicate.

2.3.5 End-Product Generation

Multiple end-products may need to be produced from the process representation; examples are guidebooks for new project members, checklists for use by experienced practitioners, and training materials for new employees.

Producing process end-products, once their goals have been defined, is largely an information management problem, not a word processing problem. Word processors principally help you only in the presentation of information, not in the management of that information. Maintaining relationships among your process information and extracting subsets of that information requires the representation to be stored in a highly-structured form.

Another aspect of the information management problem is how to quickly and efficiently produce updated process definition end-products. The goal is to avoid one of the prime reasons for unused documentation: everyone knowing that the actual practice evolved out from under the documented approach. The approach presented in this guidebook recommends and describes the use of a relational database management system to support producing multiple products from the process representation and for easily regenerating them as the process representation is better understood, refined, and updated.

2.4 PROCESS DEFINITION NEEDS

Process definition can satisfy a variety of needs. As presented in Sections 1 and 2, your goals will substantially affect how you conduct your process definition effort. Most goals can be mapped to the following five needs:

- To perform successful work
- To facilitate training
- To improve the process being used
- To engineer new processes
- To automate processes

This guidebook addresses all five needs. Sections 2.4.1 through 2.4.5 summarize the ways in which defined processes support each of these needs.

2.4.1 Perform Successful Work

An organization needs the ability to repetitively and consistently perform successful work. This ability is not only necessary for one division or group, but it may also be important for multiple divisions or groups to simultaneously perform the same type of work in the same way. Having a defined process directly supports having a repeatable process. Although work may be successfully repeated in the absence of a defined process, such repetition depends entirely on the efforts of individuals. Conversely, once a process is defined it can become institutionalized and preserved by the organization.

2.4.2 Facilitate Training

Another key use of process representations is to facilitate training. Training employees in their respective processes improves the likelihood that they understand the key characteristics of the process they should be following, which can quickly lead to higher employee productivity, efficiency, and morale. Note that it is not the process representation itself that yields these benefits, but the training opportunities that result from having a representation of the process.

2.4.3 Improve the Process

As the Department of Defense (1988) Total Quality Management Plan said, "The first step is to identify and define the processes by which work is accomplished." Engineering of processes is usually part of improvement.

2.4.3.1 Preserve Lessons

Key advantages to representing a process are the preservation of lessons learned and insights gained from performing that process. Typically, processes evolve. People involved in either managing or participating within the process often find a variety of ways to improve how they perform their work. Unless a process is defined, there is an increased risk that as individuals move to work in other areas, their process improvement insights move with them. By having a representation of a process and by updating that representation to reflect process improvements, the insights and lessons learned through performing a process are preserved.

2.4.3.2 Transfer Process

The existence of a process definition helps both to propagate a process within an organization or project and to transfer better processes in from outside. Transfer success is further aided if the definition includes introductory, training, and reference material suitable for all the roles involved in transfer, use, and evolution.

2.4.4 Engineer New Processes

Engineering of a process is applicable across any phase of a process lifespan, including its creation, improvement, and replacement. This guidebook emphasizes the definition and analysis aspects of process engineering.

2.4.4.1 Define

Obviously, definition methods and representations are central to the activity of defining processes. The material in this guidebook supports both creating a process for the first time (that is, for defining either a new process or the next generation of an existing process) and defining a process already in existence.

2.4.4.2 Analyze and Compare

Finally, one of the most important key uses of process representation is to facilitate the analysis and comparison of both existing and proposed processes. In the absence of process representations, detailed analysis of existing or proposed processes is difficult, if not impossible. All discussion on the process would be based on individual experience and opinion. Typically, there needs to be a consensus on what the process is before the focus can be turned to how the process can be improved. Without a process representation, valuable time and energy are lost while analysts attempt to reach agreement on their differing impressions of the process. By defining and creating a variety of simple models of a process, the key characteristics of the process are communicated more consistently and less ambiguously. This tends to minimize confusion about what the process is and frees the process analysts to focus on how to analyze and improve that process.

2.4.5 Automate Processes

Two motivations for automated or executable process definitions include automated support and simulation. While this guidebook version gives them limited treatment, you should be aware these are two areas with strong future potential. Automated processes can directly contribute to process analysis, planning, improvement, and training.

2.4.5.1 Automated Support

Over the last decade, a series of International Software Process Workshops has been held, mainly inspired by the potential for automated support. Although they are limited, early products are beginning to appear. These products are not only targeted to software process support but to office workflow and business reengineering. Some groupware products have potential for process support. Scripts or more advanced mechanisms are steadily advancing. These mechanisms can compose and orchestrate the execution of multiple tools, when needed, with little or no human action.

Although automated support is an area with more promise than actual products, it is still important to both your mid- and long-term planning.

2.4.5.2 Simulation

Simulation tools originally intended for other or general purpose uses have been used to try to simulate software processes. Software process simulation experience is very limited but shows some promise for engineering, management, and training. The creation of "virtual projects" with stochastic simulation allows both analysis of what may happen and case studies for training. One advantage is that the effects of the stochastic and multiple-feedback-loop nature of projects and organizations are much easier to appreciate with some of these simulation techniques.

As in all modeling of human organizations, many questions exist about the appropriate purpose, scope, and fidelity for a model to be useful. Definitions that can be modeled and instantiated to degrees needed for useful simulation will offer significant advantages in the long term. Volume 2 of this guidebook offers extensions that support the improvement of definitions over time toward increasingly rigorous and formal descriptions suitable for simulation and other automated analysis.

2.5 DOCUMENTING YOUR PROCESS USING MPDM

MPDM consists of a flexible set of templates, optional graphic notation, and various techniques for capturing and representing processes and for representing the relationships and constraints between and within the products, roles, and activities comprising the processes. MPDM provides guidance on the content of process and method descriptions and their use. In addition to presenting MPDM fundamentals, this guidebook provides a full set of examples, in Section 5 (supported by Appendix A of Volume 2), of using MPDM and its associated templates and techniques.

In general, MPDM is a template-centered methodology, combined with a goal-driven process representation strategy, so that the definitions will meet your constraints and needs. Automated support for producing guidebooks and training material from this template-based definition is central to the efficiency of MPDM, both initially and when changes are required.

Figure 2-3 describes how your process definition is used to generate different types of end-products. You use templates to gather process information and organize and verify it both for consistency and completeness. You use an accompanying graphic representation that maps from the templates and allows a better overall view of the process definition. Volume 2 discusses the option of using the templates to prepare models in other notations.

MPDM has evolved with lessons learned from an extensive pilot project (described in Section 6) and contains a number of characteristics that help make it practical. It explicitly identifies and organizes the kinds of information needed and is understandable by both management and engineers. It supports incremental improvement in both process definitions and in the process of process definition. It does not require esoteric notations or skills, yet can be used to define complex processes. Thus, MPDM is suitable for organizations just beginning process improvement and for organizations with advanced process experience and correspondingly advanced requirements.

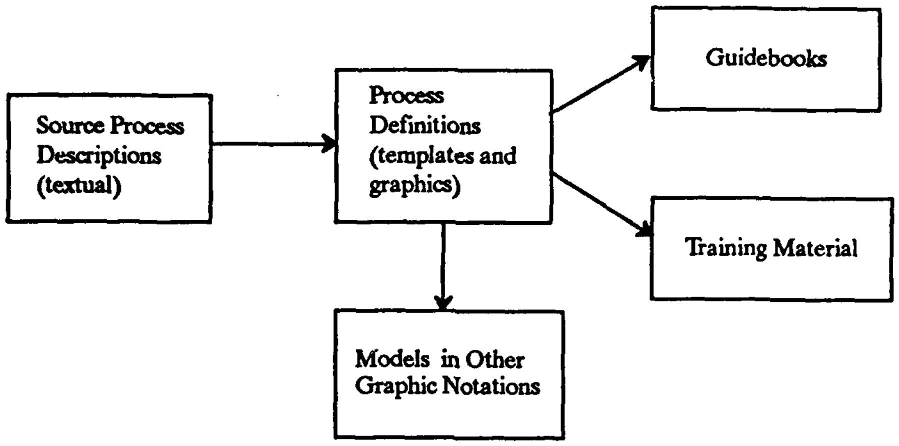

Source Process Descriptions (textual)
Process Definitions (templates and graphics)
Guidebooks
Training Material
Models in Other Graphic Notations

Figure 2-3. Kinds of Process Descriptions

This page intentionally left blank.

3. MPDM TEMPLATES AND GRAPHICAL NOTATION

3.1. OVERVIEW OF THE CONCEPTUAL MODEL

Software process definitions and models are abstractions of the software process. You can use these abstractions to answer many different types of questions about the process. Various types of models answer different questions effectively, and no single process representation can answer all the key questions about the software process. The type of model to be developed invariably depends upon the audience who will use it.

The objective of a model is to emphasize important characteristics and reduce or eliminate nonessential information based upon the needs, background, and orientation of the target audience. The principal intended users of MPDM are process engineers who are producing process guidebooks and training materials for use by people in their organization or project. The material in Section 3 specifically targets this audience.

3.1.1 Process Object Classification

To adequately define a software process, your definition must extend beyond just the activities that make up the process. You must define the types of products produced by the process; the resources required; and the relationships between products, resources, and activities. To determine what a template-based model should capture, you need to analyze the concept of a process. Figures 3-1 through 3-4 are a series of figures that motivate the conceptual model used as the foundation of MPDM and the information it captures.

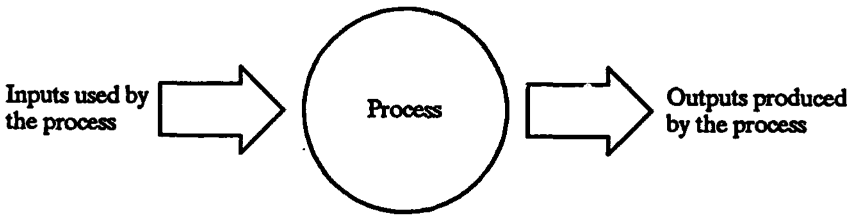


Figure 3-1. Process Model—Level 1

As shown in Figure 3-1, at the simplest or most abstract level, a process uses externally provided inputs to produce externally available outputs.

Figure 3-2 shows two classes of inputs: variable inputs and fixed inputs. Variable inputs are transformed in some manner to produce the output; they are called products. Examples of fixed inputs include machines, meeting rooms, management staff, engineering staff, and administrative staff; fixed inputs are referred to as process resources.

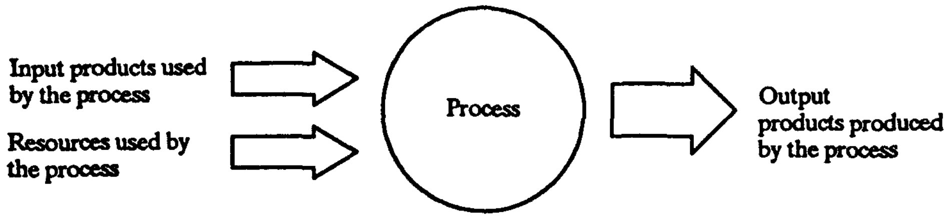

Input products used by the process
Resources used by the process
Process
Output products produced by the process

Figure 3-2. Process Model—Level 2

It is diagrammatically advantageous to separate products from process resources, because they play different roles in the process definition. In Figure 3-3, resources are at the bottom of the circle representing the process. The revised diagram implies that input products are directly related to the output products, but the performance of the process depends on the process resources.

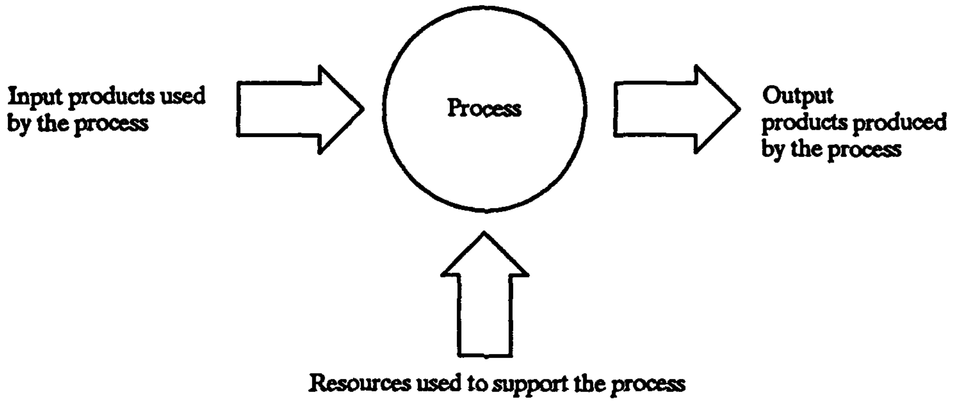

Input products used by the process
Process
Output products produced by the process
Resources used to support the process

Figure 3-3. Process Model—Level 3

At this point, the model represents a process as an activity that converts input products to output products by relying on a set of process resources. The model must also show, as indicated in Figure 3-4, that a process is a set of activities subject to control. Examples of process controls are quality standards or constraints on available resources.

The definition of a process is now: a **controlled** set of **activities** that uses **input products** to produce **output products** by relying on a set of **process resources.**

If you are familiar with the IDEF0 or SADT methods for process modeling, you will note that this perspective is analogous to the approach used in those representations. Consequently, MPDM provides a way of incrementally collecting material that you can use to create these and other graphic models.

The model shown in Figure 3-5 directly corresponds to the templates discussed in detail in this section. Discussion of constraints is deferred until Volume 2.

- *Activity Templates.* Activity templates capture information about the activities occurring within a process. They define the entrance criteria, exit criteria, and key relationships for those activities.
- *Product Templates.* Product templates are used to capture information about both the inputs and the outputs of a process
- *Support Templates.* Support templates capture information about people and mechanisms needed to support a process. Only the people aspect, or roles, are discussed in this section.

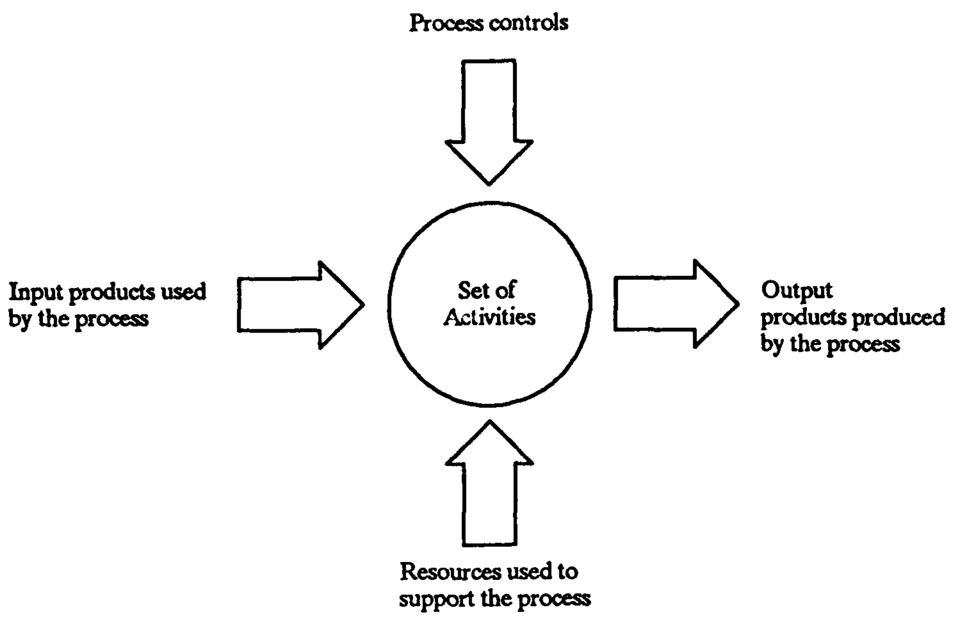

Process controls
Input products used by the process
Set of Activities
Output products produced by the process
Resources used to support the process

Figure 3-4. Process Model for the Template-Based Approach

Eventually, you will need to model risks associated with the process. Risk can take a variety of forms. As indicated in Figure 3-6, there may be risks that are related to the input products, different risks related to resources, still other risks related to various activities, etc. In short, the potential for and existence of risk surrounds all aspects of our process. (Risk templates are further discussed in Volume 2.)

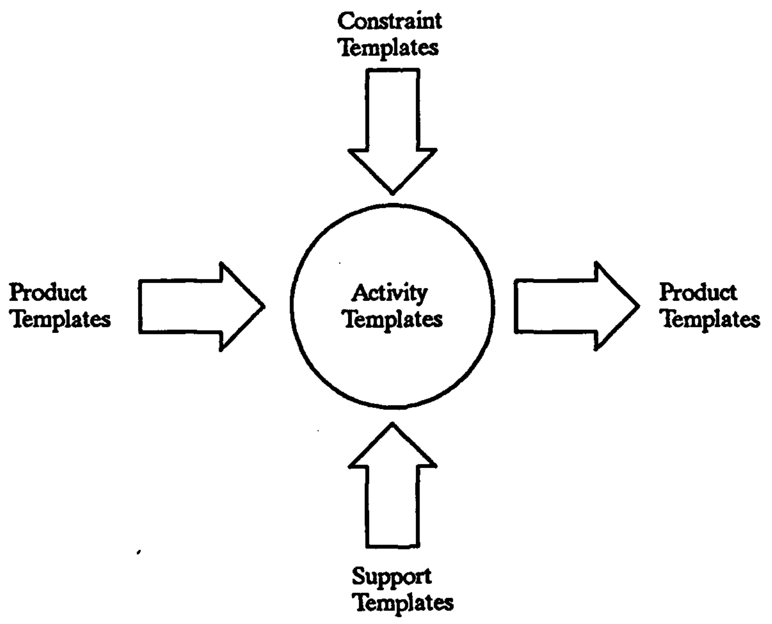


Figure 3-5. Meta-Class Templates

3.1.2 MPDM Conceptual Model Summary

To summarize, the conceptual model of MPDM is simple. A process is a set of activities that typically:

- Has inputs
- Produces outputs
- Requires resources
- Is subject to constraints
- Potentially embodies risk (see Figure 3-6)

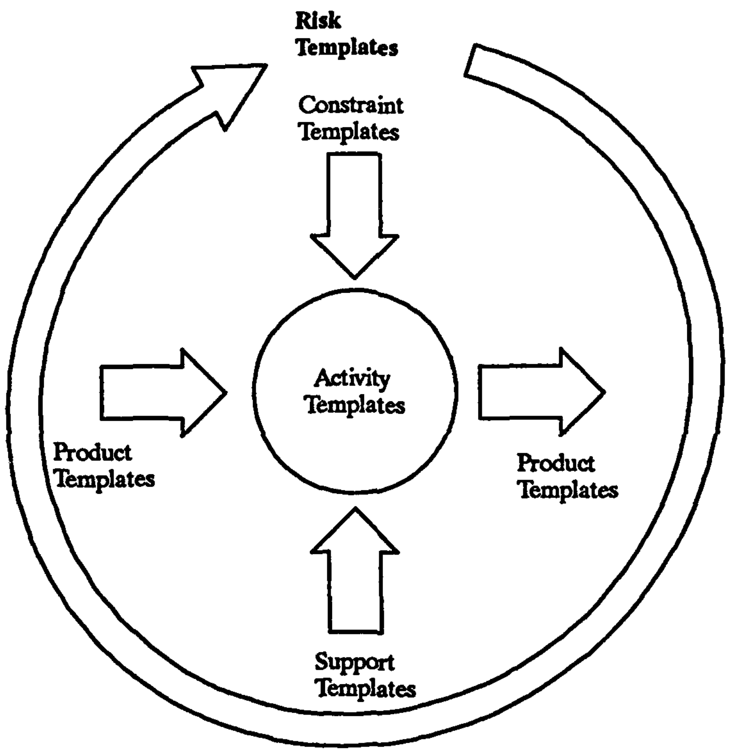


Figure 3-6. Meta-Class and Risk Templates

MPDM also assumes that multiple activities will be underway simultaneously and that activities, products, and roles can sometimes be described in terms of their state. State information is "knowable" by any activity. This convention simplifies modeling because an activity does not have to produce an output describing its state for subsequent reference by other activities. Because state information introduces complexity that is often only cost-beneficial when high degrees of formality are needed, virtually all use of and discussion of state information is deferred until Volume 2.

You can build process models that include previously defined models. For example, if a process requires use of certain configuration management procedures, the process model can incorporate a configuration management process model "by reference" on one of the templates. In this way, process descriptions that have been proven effective elsewhere in an organization can be reused when defining a project's process.

The templates and the associated graphical notation (discussed in Section 3.6) provide a simple means for capturing, manipulating, organizing, and analyzing process information. The graphical notation is used to provide a high-level view of the process and to show relationships between process information detailed on the templates. The process definition is contained solely in the templates; the graphical notation is used only to provide alternative views of the information contained in the templates. If you are using IDEF0 or another such notation, you can use the graphical notation of these methods in conjunction with the templates.

MPDM allows you to construct process models by a variety of process modeling methods. For example, if your process model is built using an object-oriented modeling technique, then the product templates and states of products will play a key role in influencing the sequence in which activities occur. If you select this approach, you will not be describing the sequence of activities in terms of product flow but will instead describe product evolution in terms of sequenced activities.

Conversely, if you prefer to use an activity decomposition technique, then activity templates and the hierarchical and behavioral relationships that exist between them, will be the primary tools in defining the process.

MPDM is not biased toward any particular process description paradigm. Object-oriented process models, functionally-oriented process models, management-intensive process models, risk-driven process models, spiral process models, etc., can each be depicted using the templates described in this guidebook. The primary differences will exist in the templates selected for use, the details allocated to these various templates, and the types of relationships established between the templates.

3.2 GOAL-BASED PROCESS DEFINITION

A common problem with process definition methodologies is that they are often "all or nothing" techniques. That is, there is essentially only one right way to apply the methodology. This results in methods that are too complex for smaller problems, but insufficiently broad for larger problems.

MPDM is highly scaleable. You can choose between using as few as 3 classes of templates, to as many as 25. You can choose between capturing, for instance, 15 or 20 different fields of information or capturing hundreds of fields. To help you in selecting the best approach, MPDM recognizes four **tiers** of process definition where each tier has different goals. Generally speaking, higher tiers of process definition require increased formality. The four tiers are:

- *Tier 1: General.* The goal of Tier 1 usage is to create and maintain process models that facilitate the production of process-related end-products that must be usable by large, general audiences. Examples include guidebooks, training material, and process-related sections of contract proposals.

- *Tier 2: Procedural.* The goal of Tier 2 usage is similar to Tier 1, except that more detail is desired. Typically, this reflects the evolution of guidebooks that convey progressively more detail. Operations manuals, for example, are process end-products that typically contain high degrees of "how-to" details.

- *Tier 3: Formal.* The goal of Tier 3 is to provide end-products that facilitate static analysis of the process. The purpose of such analysis can be to discover desirable or undesirable process characteristics, identify areas of unusually high process risk, etc.

- *Tier 4: Enactable.* At Tier 4, the goal is insights into the behavioral characteristics of a process, whether virtual or real. Currently, this is primarily achieved through automated process simulators and enactment environments, respectively.

This guidebook considers people pursuing Tier 1 and Tier 2 goals to be the principal audience. Focus, scope, and details are presented accordingly. As implied earlier, only Tier 1 process definition is described in this volume. Tiers 2 through 4 are all considered to be advanced, and are therefore discussed in Volume 2.

MPDM further characterizes the process definition effort as needing to capture and manage three primary types of important information:

- Information important for the process model
- Information important for model-derived, process end-products (e.g., guidebooks or training material)
- Information important for management and review of the process definition and modeling effort

Note that each of these information groups varies in importance as a function of role. Generally, the process engineers or SEPG members use the process model information, the audience for your end-products uses the end-product information (which augments and "packages" the process model information), and those responsible for managing the process modeling effort—which may be the process engineers themselves—use the management, work progress, and review related fields.

3.3. BASIC USAGE

Constructing process models is one of the first steps on the way toward generating reliable, consistent, maintainable process end-products. Section 4 presents the full set of steps, or tasks, in MPDM. There are numerous types of models you can build, and each is used to communicate something different. Four types of models that you will find useful are:

- *Architectural Model.* Architectural models allow you to see how your process is constructed. For example, an activity may really consist of three subactivities, and one of those subactivities may itself have several still "smaller" tasks. Think of architectural models as a type of hierarchy chart. It shows how parts of things are built up into progressively larger parts. The template fields "parent," "child," and "child type" directly facilitate constructing architectural models (see Sections 3.5.1.1.7 and 3.5.1.1.8) of the following three types of process objects:
  - Activity
  - Products
  - Roles
- *Behavioral Model.* Behavioral models convey the sequencing of events in time, and highlight process execution characteristics such as concurrent activities, alternative activities, activity rendezvous, etc. The entry criteria, internal process, exit criteria, and invariant fields are the

primary ways in which behavioral information is captured on the templates. For advanced purposes, constraint templates (discussed in Volume 2) are also used.

- *Interface Model.* Interface models communicate how activities, products, and roles are related within a process. The most common form of these models is information flow diagrams. These models are constructed by examining activity templates for the information groups that show the products needed by and released by the various activities. In the advanced templates, this information flow can be mapped directly into decision-making activities (which, in turn, are mapped to the roles responsible for making those decisions).

- *Organizational Model.* Organizational models convey how the human resources within a process communicate with each other. Note that this is different from the "authority" structure implied by an architectural model of roles (that is, levels of authority). For instance, an architectural model of a division might show that a division manager has authority over project managers, each of which has authority over several team leaders who have authority over their respective teams. However, the project managers may have to report to someone in the quality assurance group and someone in the business or finance office. In this example, the quality assurance group does not have "authority" over the project leaders, but they do need periodic reports from those project leaders. This is the key difference between organizational models and architectural models of roles. On the role template there are specific fields to capture information about who reports to a given role and to whom the given role reports.

If you are suspecting that a reporting relationship can also be conveyed using interface models showing project managers passing reports to the configuration management group, you are correct. It is always the case that a body of process information can be modeled in a variety of ways. The key issues are what are you trying to communicate and which type of model best supports that type of communication. It cannot be over-stressed that process models, with the possible exception of Tier 4 usage, are chiefly valuable as communication tools.

You can build any of these and numerous other types of models from the information gathered on the process templates. You should note that there is a general relationship between the type of models you build and the formality needed in your process descriptions. At Tier 1 (the least formal) architectural and organizational models are very important. At Tier 2, you may find it useful to include a variety of interface models. At Tiers 3 and 4, behavioral models become critical. Further details on modeling are presented in Section 3.6.

3.4. MPDM TEMPLATE INFORMATION INHERITANCE

MPDM embodies an information inheritance architecture that allows you to easily tailor or extend this methodology to suit your particular needs and preferences. However, before augmenting this methodology—and to understand the organization of material presented in Section 3.5—you need to know the following inheritance principles.

As described in this guidebook, templates have four levels of information inheritance:

- *Foundation Template.* All templates are derived from a common foundation template.

- *Meta-Class Template.* Meta-class templates inherit common fields from the foundation template.

- *Class Template.* Class templates are derived from meta-class templates and add information unique to that template.
- *Subclass Template.* Subclass templates are used to further refine and distinguish process information.

When using the templates, these four levels of information inheritance translate into two types of usage:

- *Final Template.* These are the templates actually used for collecting data. In Figure 3-7, final templates (through Tier 4 usage) are shown with a solid border.
- *Contributing Template.* These templates only contribute fields to final templates and are not actually filled out. In Figure 3-7, contributing templates (through Tier 4 usage) are shown with a dashed border.

Note: For Tier 1 usage, the final templates are activities, roles, and products; the contributing templates are foundation and supports (see Figure 3-7).

The foundation and meta-class templates are useful for discussing fields common to groups of templates and to facilitate tailoring the templates to site-specific objectives. This structure allows for easier construction and discussion of the templates and their fields. Furthermore, this approach facilitates the use of automated tools for designing, maintaining, and using electronic versions of the templates.

All templates have fields such as the following in common: Name, Unique Identifier, Description, Comments, and Revision History. Any field intended to be used on all final templates, regardless of type, is shown on the foundation template. The discussion of such globally common fields is found in Section 3.5.1, which describes the foundation template. All meta-class templates "inherit" these common fields from the foundation template. Each meta-class template adds additional information common to class templates of that meta-class and not common to class templates of a different meta-class. For example, under the activities meta-class, there are class templates for production and audit activities. These two classes have certain fields in common with each other, for example, entry criteria and exit criteria. Each class therefore provides space for a description of these criteria. Because of this principle of inheritance, any field shown at the meta-class level of a template appears on the class templates subordinate to that meta-class. All fields that the meta-class template inherited from the foundation template will also be included.

These are generalized templates and it is expected—indeed, encouraged—that you tailor them to the needs and characteristics of the corporate or organizational environment in which they are employed. Similarly, you can also tailor these templates for the actual process notation favored by a given environment. The purpose of the hierarchical architecture in the template design is to explicitly facilitate the tailoring process.

The technique for tailoring is simple. Consider the scope of any new field you want to add:

- Will the new field be used on all of the final templates? If so, put the new field on the foundation template; all other templates will then inherit this field.
- Will the new field be used on both the role and the resource templates? If so, change the support template at the meta-class level and allow the role and resource templates, and any other defined support templates, to also inherit it as a common field.

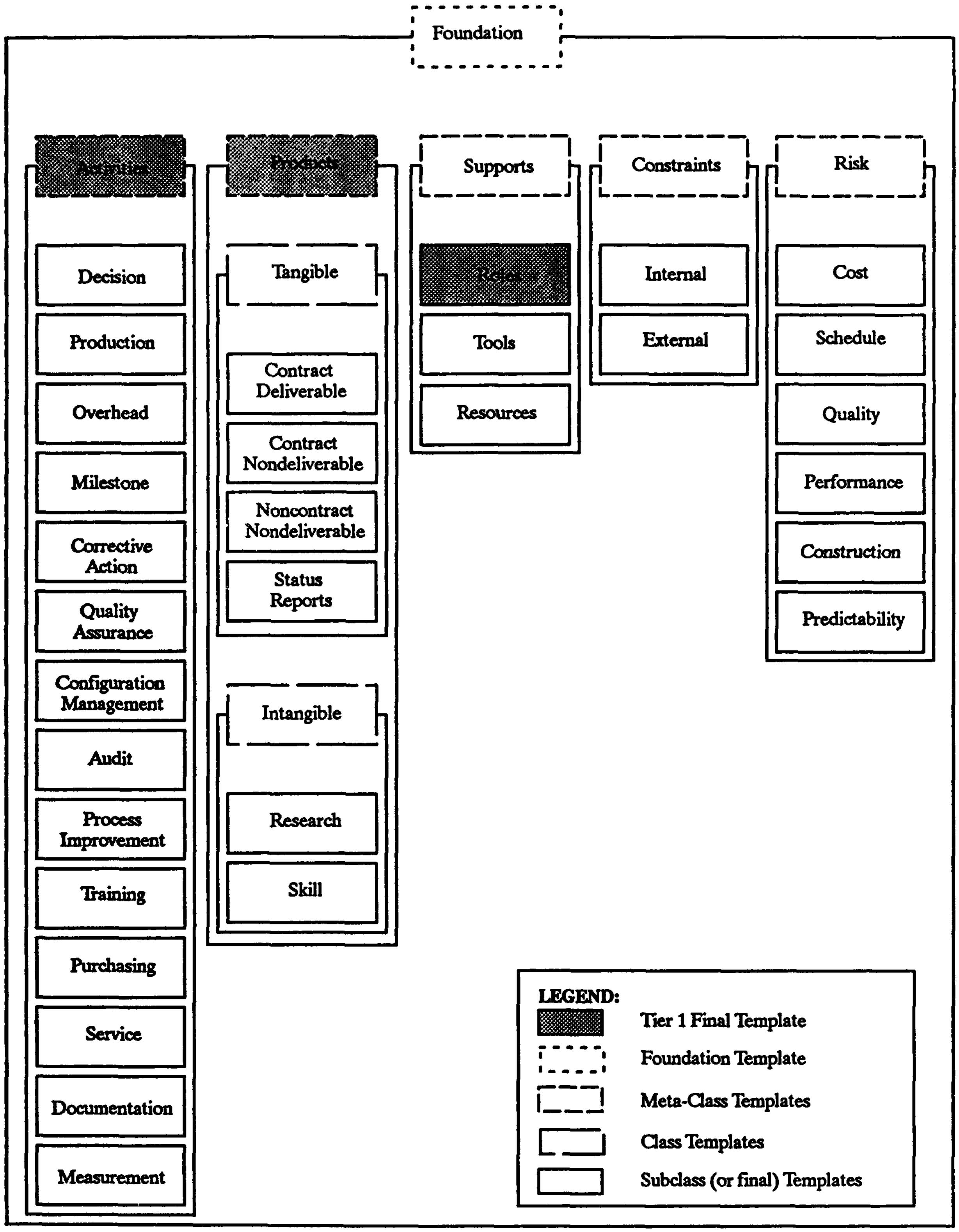


Figure 3-7. Template Inheritance Model

This approach makes it easy to introduce and document changes to the templates, yet preserves the underlying consistency between templates.

To reiterate, the entire inheritance structure of the full set of MPDM templates is shown in Figure 3-7, and only the three shaded templates (activities, products, and roles) are needed for Tier 1 usage. All other templates (Tiers 2 through 4) are described in Volume 2.

3.5. MPDM TIER 1 TEMPLATES

3.5.1 FOUNDATION TEMPLATE

Sections 3.5.1.1 through 3.5.1.3 present the Tier-1 fields on the foundation template. As discussed in Section 3.2, the type of information you need to collect is clustered into three different types of usage:

- Information used for process modeling
- Information used for generating model-derived process end-products
- Information used for management and review of the process definition and modeling effort

3.5.1.1 Model-Specific Fields

Typically, the first type of information you will collect is process information that applies directly to the process model itself. Of the fields described in Section 3.5, only a few are **required** for defining your process, the remainder are optional as a function of your goals. As you read the field descriptions, make a preliminary list of the fields that seem important to you. Keep in mind these two key considerations:

- The end-products you want to produce
- The resources (time, budget, etc.) that you realistically expect to be available for this effort

Section 4 provides specific advice on how to build and maintain process models. For now, it is sufficient for you to familiarize yourself with the field names, the purpose of those fields, and to which of the three templates (activity, product, or role) they apply.

3.5.1.1.1 Unique ID Field

Each template requires a unique identifier. It is highly recommended that you adopt the following conventions:

- *Keep the identifier as short as possible.* Identifiers will be one of the most repeatedly used fields on the templates. Shorter names reduce the amount of typing/writing required when referencing templates by their identifiers.
- *Make the identifier meaningful.* It greatly facilitates understanding if you use abbreviations to build the unique identifiers. Probably the least meaningful identifiers are those that are constructed entirely of numbers.

3.5.1.1.2 Name Field

This field is for the name of the template. Again, brief and meaningful names are best. In this field, there is no real advantage in prepending the names of ancestor templates. Indeed, doing so has the disadvantage of creating names that are too long.

3.5.1.1.3 Comments Field

This field is for notes about the information on the template; it is **not** intended for information about the process being modeled. It is quite useful, when constructing and maintaining templates, to have a place where you can make notes to yourself or to other developers about things to be done, current problems, suggestions for changes, etc. This field is intended for that usage. Think of it as a "scratch-pad" area.

3.5.1.1.4 Depth Field

This field represents the generation depth of the template. For example, in an hierarchical organization of activity templates, the activity template at the top of the hierarchy has a depth of 0, the first level under that template (or its "children") has depth of 1, any "next generation" child templates under Level 1 have a depth of 2, etc. This information facilitates orientation while quickly scanning hard-copy printouts of templates. Using an automated tool, this field can be easily calculated by the tool, and hence would not require user input.

3.5.1.1.5 Purpose Field

This field is used to provide a very brief explanation of why this process object is necessary. The motivation for having the process object (activity, role, product, etc.) is provided in this field.

3.5.1.1.6 Description Fields

There are four types of description fields, and each has a specific use. The four fields are:

- *Graphic Description.* The Graphic Description field gives you a place to include a picture of the object being represented by the template. For instance, if the process object is the Inspection activity, then you might want to include a picture which graphically depicts the various stages of the Inspection activity and how those stages relate to each other. Practically speaking, slides, such as might be used in presentations or training, should be considered "graphics" even though a slide might just contain a list of bullets.

- *Brief Description.* The Brief Description field contains a one-sentence (ideally, one-line) description of the process object. This field will be used repeatedly throughout end-products whenever there is the risk that showing just the name of a process object may fail to sufficiently inform the reader.

- *Overview Description.* The Overview Description field contains a detailed description of the process object in terms of its characteristics, content, scope, etc.

- *Summary Description.* The Summary Description field contains details about the process object. To understand how to properly use these two fields (Overview and Summary) think about exporting this information into a variety of outputs, such as guidebooks, proposals, etc. Some descriptive information you want to print **before** you print the progressively more detailed information found on any child templates beneath the current template. It is also common to want to print some summary information (hence, the field name) **after** you have printed the child template details.

Generally speaking, use the Overview Description field to provide an overview of what you are about to tell them, and use the Summary field to provide a summary of what you just told them. Detailed material, from one or more generations of child templates, fills the void between. Note that if a template has no children (i.e., it is a "final" template) then there is no need to split descriptive information between Overview and Summary, as the information will be output contiguously regardless. In such circumstances, just use the Overview field.

3.5.1.1.7 Parent Template(s) (Multiple Occurrence)

This field is a multiple occurrence field that contains the Unique IDs of the parents of the current template. In the majority of cases, any given child template will have only one parent template, but multiple parents may occasionally be useful in some areas of your process model. For instance, if you have a process object that describes how you conduct peer reviews, you may want to show that as a child under many different parent development activities.

3.5.1.1.8 Child Templates (Group of Fields)

This is a multiple occurrence group of fields. Each information group is composed of four fields. For each child, you need to know:

- *Unique Identifier (multiple occurrence).* The Unique Identifier is the identifier uniquely associated with the particular child of interest. Since there are typically two or more children, there will be a corresponding number of unique identifiers. Fields related to that child (type, tailoring, and proximity) will be grouped with each identifier.

- *Child Type.* The Child Type field contains one of two values. The child is either a **composition** child or a **specialization** child. A set of two or more composition children is used to indicate that the parent is composed of each of the children, in union. Put differently, each child is literally a part of the parent, and the parent is a composite of all of the children.

  Conversely, a set of two or more specialization children indicates that the child templates are each a special form of the parent and, depending upon circumstances, just one of the children is selected. For example, consider a process model where one of the nodes is intended to represent a compiler for a programming language. The specialized children of that template may be a template representing an Ada compiler, another template representing a C compiler, another for a FORTRAN compiler, etc. The specialized child relation tells us that just one of these compilers is selected.

  Contrast this with a node that represents an inspection team. This template might have composition child templates such as Moderator, Scribe, Inspector, and Reader. In this example, the parent is composed of all the children (the team is composed of all the team members).

  From the logical perspective, composition children are an "and" relation to the parent (the parent is made of child-1 *and* child-2 *and* child-n), whereas specialization children are an "or" relation to the parent (the parent is an instance of child-1 *or* child-2 *or* child-n).

- *Child Tailoring.* The Child Tailoring field contains information about which children are required, which are optional, etc. These tailoring options allow for plans to be built from the process model, but which are tailored to the circumstances and process drivers of particular projects. The values for this field are:
  - *Required.* Required indicates that the parent requires this particular child, and there is no option to tailor it out.
  - *Recommended.* Recommended indicates that the child is typically included in the parent (in the case of a composition child type) or is one of the most common forms of the parent (in the case of a specialization child type). However, there is a waiver procedure available where a justification for excluding or not selecting that child can be evaluated and possibly approved.
  - *Suggested.* Suggested indicates that it is within the authority of those planning or managing the project to use their own judgment to determine whether that particular child is included or selected. Generally, suggested children do become part of project plans unless there is a compelling reason to exclude them. In any event, they may be excluded at discretion, and without going through a waiver process.
  - *Optional.* Optional indicates that the particular child is **not** typically included in project plans, but that the person or persons planning or managing the project should be aware that an option to use the child exists, and they may, at their own discretion, elect to choose or include that child.
  - *Restricted.* Restricted indicates exactly the opposite of Recommended. If you want to include this child as part of your project plan, you will have to go through a waiver process that exempts you from the policy restricting the selection of this child.
  - *Prohibited.* Prohibited indicates that the child is not to be included (if it is a composition child) or selected (if it is a specialization child). No waivers exist. Although at first it seems unusual to define prohibited process objects, the concept becomes important as your process assets become more abundant and tailorable. For example, as a function of different process drivers, an activity that is recommended in one life-cycle model may be prohibited under a different life-cycle model. A process object would never be prohibited under all circumstances.
- *Child Proximity.* The Child Proximity field contains one of two values. Proximity is either **local** or **remote.** Local proximity indicates that the information about that child is stored locally and, hence, is immediately available. Remote proximity indicates that although further information about the child is available, it is elsewhere (such as in a completely separate database) and, hence, not immediately accessible. The premise is that another template or set of templates exists elsewhere which further elaborates the nature and content of the child.

3.5.1.1.9 Transitions (Group of Fields)

This group of two fields contains information about process objects that fundamentally change to different objects within a given model. The two fields in this group are:

- *Changed From/Unique ID.* The Changed From/Unique ID field indicates the unique identifier of the template that represents what the object used to be.

- *Changes To/Unique ID.* The Changes To/Unique ID field contains the unique identifier of the template that contains information about what this process object will eventually become.

These types of metamorphoses are quite rare within process models. Usually it only occurs when products later become constraints. For example, a design activity might create, as an output product, a detailed design document. However during coding, it is typically more accurate to show the detailed design document as a constraint and not as a product. Note that if template-a indicates it changes to template-b, then template-b must correspondingly indicate that it changed from template-a. Typically, this type of bidirectional integrity can be automatically enforced in a database environment.

3.5.1.1.10 Additional Information (Group of Fields)

This is a multiple occurrence group of fields that contain information about where additional material about this process object can be found. This includes the source material from which the information on the template was derived. These fields may also contain information about where someone should look if they are seeking additional details (which may or may not also be available in other parts of the process model). This group is composed of the following fields:

- *Title (multiple occurrence).* The Title field contains the name of the reference document.

- *Section (multiple occurrence).* The Section field contains further information to facilitate someone easily finding any relevant information. Section is a repeating field under title (that is, for a given title, there may be multiple sections that have relevant information), and pages is a repeating field under section.

- *Pages (multiple occurrence).* The Pages field is intended to carry a range of pages, and would be used as a repeating field only if, for a given section, the reference information was in two or more groups of pages (for example: Section-N pages 12–14, pages 21–25, page 36, and pages 4–51).

3.5.1.2 End-Product Fields

All the fields described in Section 3.5.1.1 are directly useful for modeling your process. However, you also have the need to capture information that makes the model information more usable in various end-products. The fields in Section 3.5.1.2 contain information that is specific to creating usable process end-products from the model-specific information.

3.5.1.2.1 Used Within (Group of Fields)

This is a multiple occurrence group of fields that contain information about where the template information is currently being used or will be used and whether it can be used on the next build of that product. Specifically, this group of fields is composed of:

- *Output Object Name (multiple occurrence).* The Output Object Name is the name of the intended output artifact or product. This includes guidebook names, the names of training courses, proposal names, etc. Special care must be taken that the same output product is always referred to by the same name. That is, the name must always be treated as a unique identifier.

- *Output Object Type.* Output Object Type indicates the type of object within which the information will be used. This includes guidebooks, training courses, proposals, technical reports, operations manuals, project process manuals, etc. For the purposes of automation, the information for this field should be provided by the environment as a function of the Output Object Name. That is, to ensure consistency, it is recommended that you construct a file of output objects that contains both their name and their type.

- *Include In Next Build.* The field Include In Next Build is a field with three options: Yes, No, and Unknown. During the development of process models and parallel development of process-related guidebooks, etc., it is recommended that you repeatedly print incremental drafts to verify the scope, integrity, and quality of your work. This field allows you to easily indicate when you think a template has sufficient information and is sufficiently accurate to start being included in builds of output products.

3.5.1.2.2 Transition Text (Group of Fields)

This is a multiple occurrence group of fields that contain material which facilitates the presentation of process information in the various end-products. This group of fields consists of:

- *Output Object Name (multiple occurrence).* The Output Object Name field is identical to and used in the same way as explained in Section 3.5.1.2.1, Used Within group of fields. As a rule, it is illogical to have transition text for an end-product that this template is not "used within." Consequently, the Output Object Name in this group of fields must match one of the Output Object Name fields in the Used Within group of fields. However, it is perfectly acceptable, even common, to have output-objects listed under Used Within, but for which there is no transition text.

- *Leading Transition Graphic.* The Leading Transition Graphic field is available for optionally storing a graphical image that depicts important information relative to the process object. As with all references to graphics, this term is used to include any form or presentation of information that is targeted for output which is potentially graphical in nature. This includes, for instance, slide presentations—even though such presentations may consist of numerous slides which contain only text.

- *Leading Transition Text.* Leading Transition Text field contains textual information intended to improve the understandability or usability of the end-product (but, again, this is information which you do not need in the process model itself). Leading transition text is extracted **before** any other fields (such as the descriptive information) are taken from the template.

- *Trailing Transition Text.* The Trailing Transition Text field contains textual information intended to improve the understandability or usability of the end-product (but, again, this is information which you do not need in the process model itself). The trailing transition text is extracted **after** any other fields (such as the descriptive information) are taken from the template.

- ***Trailing Transition Graphic.*** The Trailing Transition Graphic field has the same purpose as the Leading Transition Graphic field, but is used when you want the graphic to be presented after all the other transition and process information on the template.

Think of the mechanics behind the goal of automatically generating end-products from the process model. You need to algorithmically navigate the model and extract information so that the information is output in the sequence necessary for the intended end-product. Regardless of how powerful and intelligent the search and selection algorithms are, when you actually read the output—guidebook or training material, for instance—you will likely find places where the transitions between topics are awkward, jarring, or confusing. To improve the readability of that end-product, it may be necessary to add some transitional text which smooths the flow of topics or information or provides content information useful to the reader or user of that product. To accommodate this need, you go to the template whose information is presented in that part of the end-product, and add the necessary transitional material. That is, *if you need a picture or some text (or both) before the process information on that* template, use the leading transition fields; if you need a picture or text after the process information, use the trailing transition fields.

It must be stressed that the purpose of these transition fields is to capture information that is **only** important with regard to the specific end-product. For example, you may want to include a Forward at the front of a particular guidebook. The question is, where do you put such information? This is information that is not part of the process per se, but is still useful to have when building a guidebook derived from the process model. If you need to include such information, use transition fields to capture it.

Because transition material is entirely end-product specific, on any given template you may have as many groups of transition fields as you do end-products.

In principle, you can write an entire guidebook, proposal, set of training material, etc., using nothing but the transition fields. You can get all the text to appear exactly in the order you want it and every chart, graph, table, and picture to be placed within that text at exactly the locations needed. However, in practice, you want to achieve this goal by *minimizing*, not maximizing, your use of the transition fields. Think of the transition fields as a form of insurance. The goal is to build a process model, populate it with information, and extract that information in ways that create completed guidebooks, training materials, etc. To some degree, you will improve those products by improving the structure and content of the model from which they are derived. However, at a certain point you will find that the model is fine and the information in it is adequate; the problem is with the characteristics, use, and requirements of the specific end-product. This is precisely the time to turn your attention toward using the transition group of fields.

3.5.1.3 Management-Specific Fields

The last type of information you need to consider collecting is information related to managing, reviewing, and assuring quality, as the work of process definition proceeds. The following fields are intended specifically for information that helps perform process definition efficiently and effectively. As a subtle distinction, it is not meant to imply this information is only of use to managers. The fields are useful in managing the effort. Paradoxically, the use of these fields reduces the burden on management personnel to closely watch the process definition effort because this information directly facilitates "self-management" by those performing the process definition work. Of course, this also provides management the means to easily and quickly evaluate progress achieved to date.

3.5.1.3.1 Requirements Traceability (Group of Fields)

Arguably, this group of information could be considered process model information but, from a different perspective, the fact that a process model needs to depict a process that complies with some set of requirements (such as ISO-9000, or requirements derived from the CMM, etc.) is typically something that management elects to impose on the process.

This group consists of the following fields:

- *Requirements Document Title (multiple occurrence).* The Requirements Document Title field is used for the name of the requirements document, policy document, standard, etc., which requires this particular process object. As with titles of other external documents, be careful to treat this field as a unique identifier. Hence, no two external documents should have exactly the same name, and any given external document should always be referred to by just one name.

- *Requirement Unique ID.* The Requirement Unique ID should be taken from the original source document or material. This unique identifier allows easy traceability between the process model and one or more requirements documents or standards. In some cases, it is useful to add extensions to the original requirements identifier. For instance, if the source document has requirement-x that is explained over 5 pages, then you may find it helpful to make traceability references to: requirement-x, paragraph 5; or requirement-x, paragraphs 11 through 15.

- *Level of Satisfaction.* The Level of Satisfaction field contains one of the following values to indicate to what degree that particular requirement is satisfied by this particular process object. The levels are:

  - Very Low
  - Low
  - Moderate
  - High
  - Very High
  - Unknown

  A single template might show that it satisfies one requirement to a moderate degree, another requirement to a very high degree, and a third requirement to a low degree. Additionally, a number of templates may all be involved in satisfying a particular requirement, and though individually each template's satisfaction of the total requirement may only be moderate, taken all together they might offer complete coverage and hence, as a collection, yield a very high degree of satisfaction. Consequently, a parent template can show very high satisfaction even though none of its children showed more than moderate satisfaction of the same requirement. Unless you have a means for quantitatively analyzing level of satisfaction, you may want to use an even simpler scale. A scale of Low, Medium, High, and Unknown will likely suffice in many circumstances. Regardless of the scale you select, always include the option Unknown, and use it as the default field value.

3.5.1.3.2 Assigned To (Group of Fields)

This is a multiple occurrence group of fields and is used to name the person or persons who are responsible for the template. This group is composed of the following two fields:

- Name/Unique Identifier *(multiple occurrence)*
- Group/Unique Identifier

As with other name or title fields, these names are considered to be unique identifiers and should be treated accordingly. At Tier 2 and higher, both fields may contain the unique identifiers of role templates. In that case, only the Name field should be directly input, the Group identifier would be found in the Parent field of the role template. In the relatively rare case of multiple parents, then the Group field must match one of parents. In Tier 1, you typically will just type in this information.

3.5.1.3.3 Work Status (Group of Fields)

This group of fields facilitates planning, allocating, and tracking the work involved in developing the process model and its ability to generate finished (or semifinished) end-products. The fields are:

- *Percent Complete.* The Percent Complete field is used to carry a value between 0 and 100 where 0 indicates that essentially no work has yet been done on the template and 100 indicates that, with regard to the current effort, the work on this template is complete. Note that if on some future date there is a decision to make major extensions in the details the model, one of the first steps would be to downgrade the Percent Complete field to reflect how complete the templates are with respect to the new objectives.
- *Planned Information Capture Start Date.* Planned Information Capture Start Date field reflects the starting time of the effort in which, according to the plan, the content of the template will be entered. This effort includes time involved in research, conducting interviews to collect information, etc.
- *Planned Information Capture Completion Date.* The planned Information Capture Completion Date field reflects the completion time of the effort in which, according to the plan, the content of the template will be entered. This effort includes time involved in research, conducting interviews to collect information, etc. The completion date is intended to reflect that period in time when the template is expected to contain a sufficient amount of information to be usable within the process model in general, one or more end-products that are to be generated by the model, or both (depending on the goals of the current effort).
- *Actual Information Capture Start Date.* The Actual Information Capture Start Date field reflects when work really was started.
- *Actual Information Capture Completion Date.* The Actual Information Capture Completion Date field reflects when work really was completed. Again, the beginning of any major effort to update the templates (or a subset thereof) would include resetting all the actual capture dates back to null values in preparation for the next wave of work.
- *Template Development Status.* The Template Development Status field is used to indicate both to management and to others participating in, or interested in the progress of, the process definition effort. Each template is given a current status from the following list:

- *Awaiting Development.* Awaiting Development indicates that although the template exists, no one has yet begun to work on it. Note that two aspects of development status can be inferred from another field. Specifically, if the Assigned To field is empty, then the template is not only only awaiting development, it is also awaiting assignment to someone or to a group of people. If that field is not empty, the template is has resources assigned to work on it, but that person or group has not yet begun the work.
- *Under Development.* Under Development means that work has begun and is underway.
- *Work Temporarily Suspended.* Work Temporarily Suspended indicates that although work was started, it is currently in a state of suspension (so do not look for anything new until the status changes).
- *Finished Incomplete.* Finished Incomplete means that although more information could be added to this template, no further additions are expected at this time. This might occur when time or funding for an effort is running out, and work is being redirected to concentrate on high-benefit parts of the model.
- *Finished Complete.* Finished Complete means that work on this template is finished and the content is sufficiently complete for the current usage.
- *Finished to External.* Finished to External means that work is complete, and valuable information is available (typically in hard-copy form) which is not part of the process model. If this is the status of a template, then there must be at least one entry under the Additional Information Available In field to tell the user where to look for the external information.

3.5.1.3.4 Version Number

For configuration management purposes, each template needs a version number in addition to its unique identifier. It is highly recommended that you include the date a version is completed as part of the version numbering scheme. This greatly facilitates reviewing prior versions of a template as a function of when the work or updates were performed.

3.5.1.3.5 Revision History (Group of Fields)

Revision history provides information about the evolution of a template and the information it contains. This group of fields includes:

- *Revision Date (multiple occurrence).* The Revision Date field contains the date that a particular revision was completed. Consider this to be similar to a release date for the template. It is highly recommended that this not be the date the work is completed, but is instead the date when the template successfully passes your quality review process.
- *Resulting Version Number.* The Resulting Version Number is the new version number assigned to the template after updates have been made, reviewed, and approved. The most recent Resulting Version Number should, of course, match the Version Number field.
- *Update Comments.* The Update Comments field allows the person or persons who updated the template to provide brief comments on the nature of update, why the update was done, etc.

- *Revised By (multiple occurrence).* The Revised By field contains the name of the person or persons who created a particular revision. Note that for any single instance of a revision date and resulting version number, there may be two or more people who participated in that revision. Again, be careful to treat this field as unique keys; that is, always refer to the same person by using exactly the same spelling.

3.5.1.3.6 Template Review Status

MPDM presumes a regular review process as part of building and maintaining your process asset library. This field and the following group of fields (Section 3.5.1.3.7, Review History) are intended to support the review process. There are five different values for this field:

- *Review Needed.* Review Needed is the default value for this field. Generally, whenever a template is created, or whenever work begins on updating or modifying the content of a template, the template review status is set to Review Needed.

- *Ready for Review.* Ready for Review indicates that the person or persons working on the template consider it to be sufficiently complete and accurate to be reviewed. This does not necessarily imply that they have stopped working on or updating the template.

- *Reserved for Review.* Reserved for Review indicates that the template is now "owned" by the review process. At this point, no further changes can be made to the template until it is released by the review group (i.e., until this field has been set to Review Completed). In an automated environment, the template should be locked, or permissions changed, to prevent inadvertent updates.

- *Under Review.* Under Review indicates the template is actually moving through the review process. As with Reserved for Review, the template continues to be owned by the review process, and hence no updates, alterations, changes, etc. are allowed.

- *Review Completed.* The review status of the template is set to Review Completed when those involved in the review process are ready to release the template back to the developers (or to those responsible for configuration management). Note that Review Completed implies nothing about the results of the review. That information is conveyed in the Review History group of fields.

3.5.1.3.7 Review History (Group of Fields)

Review history fields are used to collect information about the occurrence and results of reviews. This group includes the following fields:

- *Review ID (multiple occurrence).* Review ID contains a unique identifier that allows you to explicitly distinguish one review effort from another.

- *Review Date.* Review Date contains the date in which the review was completed.

- *Version Reviewed.* Version Reviewed indicates the version number of the template under review. Note that a single version number may be subject to more than one planned review, such as a preliminary review and a final review, and may also be subject to one or more unplanned reviews, as would occur if the template failed to pass the review process.

- ***Review Composite Result.*** Review Composite Result contains one of the following values:
  - ***Passed.*** Passed indicates the template passed the review process without any moderate or severe issues being identified by the reviewers. If any trivial issues are identified, these are passed back to the developers so that they may update the template accordingly, but no further review is needed. If this review cycle is part of a baseline evaluation, then once passed the template would be baselined before being released back to the developer(s).
  - ***Passed Marginally.*** Passed Marginally indicates that one or more nontrivial issues should be addressed by the developers. These are suggested areas to be addressed, and it is up to the developers whether they perform the rework. Regardless of whether the developers elect to address the issues, no further reviews are required to pass this phase of work.
  - ***Passed Conditionally.*** Passed Conditionally indicates that one or more nontrivial issues must be addressed by the developers before the reviewers will pass the template. However, this classification indicates that once the required rework is done, there is no need for a follow-up review. This does not imply that rework is not verified by, for instance, an inspection moderator. This only indicates that another review by the entire review team is not anticipated nor expected.
  - ***Failed Marginally.*** If the reviewers decide that rework needs to be done, and due to scope or other factors the reworked template needs another review, then they classify the template as either Failed Marginally or Failed Significantly.
  - ***Failed Significantly.*** The result of Failed Significantly implies that, in the reviewer's opinions, the template was not ready to be reviewed and needs substantial work before another review is requested.

3.5.2 Activity Templates—Tier 1

Until this point, this guidebook has discussed universal information inherited from the foundation template. This section begins to talk about fields of information that specifically apply only to a single type of template: activity templates.

There are only two types of information that need to be captured on activity templates: model-specific information and end-product related information. Since management-specific information is invariably applicable across all templates, it exists entirely on the foundation template and is, therefore, a part of every template. For activity templates, all the fields in Section 3.5.2 are model-specific with the exception of Elaboration Text fields in the Related Products and Related Supports groups of fields. Elaboration Text fields are end-product specific.

3.5.2.1 Activity Criteria and Process (Group of Fields)

The Activity Criteria and Process group of fields provides information about when the activity may start, the tasks or steps involved in the activity, when that activity may end, and conditions that must remain true for the activity to remain active. These fields are:

- *Entry Criteria.* The Entry Criteria field explains the criteria that must be satisfied before an activity may begin. Typically, this information conveys that other activities must first be completed, or certain products and resources must be available. However, as this is a text field, there are no restrictions on the type of information conveyed nor on the manner in which it is conveyed. The goal is to explain as clearly and succinctly as possible, using natural language, the circumstances which allow the given activity to begin.

- *Internal Process.* The Internal Process field explains what the activity consists of and, at lower levels in the process model, how the activity is actually performed. Again, this is not a formal description, but is instead a succinct natural language description of the activity. This field may become quite lengthy. If so, consider the advantages of establishing two or more child templates and moving some of the detail information to them.

- *Exit Criteria.* The Exit Criteria field explains the criteria that must be satisfied before the activity is considered complete. In the vast majority of cases, this will be in terms of finished products having been produced, inspected or reviewed, and, perhaps, baselined and placed under configuration management.

- *Invariants.* The Invariants field is used to capture information about things that must remain true for the activity to continue. Invariants differ from entry criteria in that entry criteria are required to be true only at the time the activity starts, whereas invariants must remain true throughout the activity. Examples of invariants: "Weekly status reports must be compiled and archived" or "Research continues only until the allocated funding is exhausted."

3.5.2.2 Related Products (Group of Fields)

For each activity there will typically be one or more products needed or produced by that activity. This group of fields captures product-related information and is composed of the following fields:

- *Unique ID (multiple occurrence).* The Unique ID field contains the unique identifier of a product template.

- *Usage.* The Usage field provides information about when a particular product is needed by an activity. The values of this field are as follows. These three options give a very general indication of when a product must be made available to or exist within a given activity.

  - *Before Starting.* Before Starting indicates the product is an input, making that product part of the entry criteria for the activity.

  - *While Underway.* While Underway simply indicates that the activity can start without the product being available, and may end without the product, but that sometime within the span of that activity, the product is necessary.

  - *Before Ending.* Before Ending indicates that the product must exist before the activity can be considered done, thereby making that product part of the exit criteria for the activity.

- *Impact.* The Impact field conveys the primary impact this activity has on the product. This field contains one of the following four values:

- *Left Unchanged.* Left Unchanged indicates that although the particular product is needed, it is not altered by this activity. As an example, an inspection process needs the artifact or product to be inspected, and then releases it back to the author for any necessary updates (hence, changes occur outside the inspection process).
- *Created.* Created indicates that this activity does not expect the product to be provided by prior work, but instead creates that product, or at least the first parts of that product, itself. Although a product that has no subproducts, or children, will only have one activity which creates it, a complex product (with, possibly, several layers of subproducts) typically will be created in a variety of activities.
- *Changed.* Changed indicates that a version of the product was passed into this activity, and this activity updates, alters, extends, or otherwise changes the product before the end of the activity.
- *Destroyed.* Destroyed indicates that the product will be dismantled, consumed, disposed of, or otherwise made unavailable to all other activities within the process.
- *Internal.* Note that at higher levels of abstraction, a product that was created in one child might be changed in another child, and destroyed in yet another child. In such cases, the value of **Internal** indicates that entire existence of the product occurs inside of this activity.

When the same product is used differently by the subactivities of an activity template, use the following conventions:

- If usages include both Left Unchanged and Changed, then at the next higher level indicate usage as Changed.
- If usages include both Created and Changed, then use Created.
- If usages include both Changed and Destroyed, then use Destroyed.
- If usages include both Created and Destroyed, then use Internal.

Also note that there are logical relationships that exist between the values of the Usage and Impact fields. For instance, it is illogical if the Impact field indicates a product is created, but the Usage field indicates that product is needed before starting. Likewise, it is illogical for Usage to require a product to be in existence for the activity to end, with Impact indicating that the product is destroyed as part of the activity.

- *Tailoring Options.* The Tailoring Options field contains information about whether the product is, for instance, required or optional within this activity. These tailoring options allow for plans to be built from the process model which are tailored to the circumstances of particular projects. The values for this field are as follows. The meaning of these terms is consistent with their usage in Section 3.5.1.1.8. However, the relationship now specifically binds activities and products.
  - *Required.* Required indicates that the activity requires this particular product, and there is not an option to tailor it out. Note that the usage field contains information

about whether the product is required before the activity can begin, while the activity is underway, or before the activity can be considered done.

- *Recommended.* Recommended indicates that the product is typically expected, used within, created, updated, or otherwise involved with the activity, but that there is a waiver procedure available where a justification for excluding that product from the activity can be evaluated and possibly approved.
- *Suggested.* Suggested indicates that it is within the authority of those planning or managing the project to use their own judgement about whether that particular product is important to their use of the activity within a particular project. Generally, Suggested products are included in project plans unless there is a compelling reason to exclude them. However, they may be excluded at discretion and without going through a waiver process.
- *Optional.* Optional indicates that the particular product is **not** typically included in project plans, but that the person or persons planning or managing the project should be aware that these product options exist. Those persons may, at their own discretion, elect to include one of more of these optional products.
- *Restricted.* Restricted, in this context, is the opposite of Recommended. If you want to produce or use this particular product, then you will have to go through a waiver process to be exempt from the policy that states not to use or make the product.
- *Prohibited.* Prohibited indicates that the activity is not to use or produce this product. No waivers exist.

• *Elaboration Text.* The Elaboration Text field allows you to describe how the particular product relates specifically to this activity. Although the product template will contain general information about the product itself, the importance or application of that product within various activities will be different. Any important product-related information not already reflected in other fields is placed in the Elaboration Text field.

3.5.2.3 Related Support (Group of Fields)

In the basic templates, the Related Support group of fields is used to capture the roles involved in performing the activity. This group of fields is composed of:

• *Unique ID (multiple occurrence).* The Unique ID field contains the unique identifier of a role template.

• *Usage.* The Usage field indicates when a particular role is needed by an activity. The values of this field are exactly the same as explained in Section 3.5.2.2 (Before Starting, While Underway, and Before Ending). However, the only time a role would be shown as not being needed until Before Ending is if that role is only involved in some type of quality assurance, sign-off, or other capacity that occurs only when, as far as the developers are concerned, the work is done and the activity is ready to end.

• *Tailoring Options.* The Tailoring Options field is described in Section 3.5.2.2, and contains the possible values of Required, Recommended, Optional, Suggested, Restricted, and Prohibited.

The use of these terms is consistent in both groups of fields, except that here you are indicating whether the role is required, optional, etc., with respect to the activity.

- *Elaboration Text.* Similar to Elaboration Text field for the Related Products group of fields, the Elaboration Text field in this group allows you to state the particular responsibilities of the indicated role. General role information should be captured on the role template; this field is intended only to capture how the role specifically applies to or participates in this activity.

3.5.3 Product Template—Tier 1

Product templates contain information about the characteristics, purpose, and use of products within the overall process model. In addition to the fields inherited from the foundation template, the product templates add a few fields that capture details about products in particular and the activities to which these products are related.

3.5.3.1 Related Activities (Multiple Occurrence)

This field contains the unique identifiers of all activities to which the product is related. A product is related to an activity if the activity needs, uses, references, alters, or otherwise expects the availability of that product in any way.

3.5.3.2 State Information (Group of Fields)

These fields are primarily of interest only if you plan to use a relatively formal graphical notation, such as IDEF0, to portray your process information. Until you have a particular need to reference state information, there is little reason to attempt to define it. Conversely, it is often convenient, particularly in entry and exit criteria fields, to be able to refer to a product as being in a particular state. When you establish and define a set of states for a product, it becomes far easier for everyone to use a common vocabulary when referring to the various transitions that a product typically goes through. This group of fields is composed of:

- *Label (multiple occurrence).* The Label field is, for all intents and purposes, the unique identifier which is the name of the state.

- *Description.* The Description of the state is a brief explanation of the characteristics or boundaries of that particular state.

The following list is an example of types of states you might want to define for a particular product:

1. Planned
2. Scheduled
3. Authorized
4. Enabled
5. Under Development

   a. Phase 1 Completed
   b. Phase 2 Completed
   c. Phase N Completed

6. Ready for Review
7. Reserved for Review
8. Under Review
9. Review Completed
10. In Rework
11. Completed
12. In Suspension
13. Cancelled
14. Approved
15. Released

And possibly...

16. Fielded
17. Planned for Retirement
18. Scheduled for Retirement
19. Being Retired
20. Retired

The purpose, use, and development of products vary widely as a function of the product itself, so it is highly unlikely that you will find a single set of states that works equally well for all the various types of products you represent in your model. Consequently, it is useful to allow for different sets of state information, where each set is applicable to one or more different classes of products.

However, it is worth repeating that the effort to identify and define state information should only be made when you are confident that this information has sufficient value and will actually be used.

3.5.4 Role Templates—Tier 1

Role templates are used to collect information about the roles people will take when performing the process. As with all templates, role templates include fields from the foundation template. Additionally, role templates, capture the activities in which a role participates and the reporting relationships of that role.

3.5.4.1 Supported Activities (Multiple Occurrence)

This field contains the unique identifiers of all the activities which need, expect, optionally utilize, or otherwise reference this role.

3.5.4.2 Reports To (Group of Fields)

This group of fields reflects those roles to which this particular role reports. This information is composed of:

- *Unique ID (multiple occurrence).* The Unique ID field contains the unique identifier of another role template.
- *Elaboration Text.* The Elaboration Text field contains clarifying information about this particular reporting relationship, such as what is reported, how often, etc.

It should be noted that in virtually all role structures, there is an implicit reporting path from child templates to parent templates. However, it is highly recommended that you explicitly capture that reporting structure using the Reports To fields (even though it is implied by the Parent field). The Reports To fields are also used to define any reporting paths not implied by parent/child relationships.

At higher levels of abstraction, you will find that there are times when this field is "not applicable." For instance, if you model an inspection team as a role, then typically the entire team does not report to someone. Instead, the team reports to the moderator, and the moderator has reporting responsibilities outside of the team.

3.5.4.3 Reported To By (Multiple Occurrence)

This field shows the other half of the reporting relationships. Specifically, it contains the unique identifiers of any other role templates which report to this role.

As explained in Section 3.5.4.2, at higher levels of abstraction there are circumstances where the role does not represent a person, and the concept of being reported to is not applicable. There are also exceptions to this. For example, "Board of Directors" can be modeled as a "complex" role, with the "CEO" role reporting to it.

3.6. GRAPHICAL CONVENTIONS

Regardless of how careful you are in collecting and distributing information across the process templates, it quickly becomes impossible to remember all the information you have gathered, where you have placed it, and how you related it to other information. Graphical models are the primary means for restoring conceptual or high-level understanding. MPDM distinguishes four different types of process models. The graphical conventions described in this section are intentionally simple. However, they do allow you to quickly and easily develop, analyze, and compare various models of your process.

It is strongly recommended that you use this or a similarly simple and fast approach while iterating through the creation and definition of your process assets. Once you are satisfied that your representation of the process is sufficient, you can export the relevant information from the database and use it as "worksheets" for developing graphical models that are more complex, labor-intensive, etc. This is typically more cost-effective than trying to maintain large, complicated, diagrams inside a difficult tool, especially during the early stages of process definition. During this time, increased understanding of the nature of and relationships between process assets results in repeatedly needing to easily make significant changes to your process models.

Another advantage to the conventions described here is that each template is represented by a node on a diagram, and each graphical node corresponds to one template. This one-to-one relationship allows you to readily confirm consistency between your database of template information and your graphical models of that information.

3.6.1 General Graphical Conventions

These graphical conventions emphasize the fact that there are process objects within your model, and there are relationships between those objects. At Tier 1, the three classes of process objects you need to diagram, and the symbols used to represent them are:

- Activities ■
- Products ●
- Roles ◆

The four types of relationships are shown as:

- Behavioral ———→
- Architectural - - - - - →
- Interface ·———→
- Organizational Reporting — · —→

These objects can be related to each other in a variety of ways. Relationships are shown using a variety of different lines—each indicating a different type of relationship. Sections 3.6.2 through 3.6.5 discuss details on the appearance and meaning of these relationships. Different lines and different arrow types are used so that you can build composite models that mix the type of information being communicated. However, there are disadvantages to doing so, as discussed later in Section 3.6.6, Mixing Models.

There are two other global conventions. First, any line that is struck by a small arc at either the "source" or the "destination" end of the line indicates an optional relationship. For example, if a line is pointing from a role to an activity, it indicates the role is needed by that activity. However, if a small arc crosses that line by the arrowhead, it indicates the role is optional. If you have, for instance, several activities under a parent activity, and one or more of the child activities is optional, it usually creates less clutter to show the arc(s) at the source end of the line as it leaves the optional child(ren). You will see several examples of this in Section 3.6 and in Section 5.

3.6.2 Behavioral Relationships

Figure 3-8 shows an example of a process behavior model.

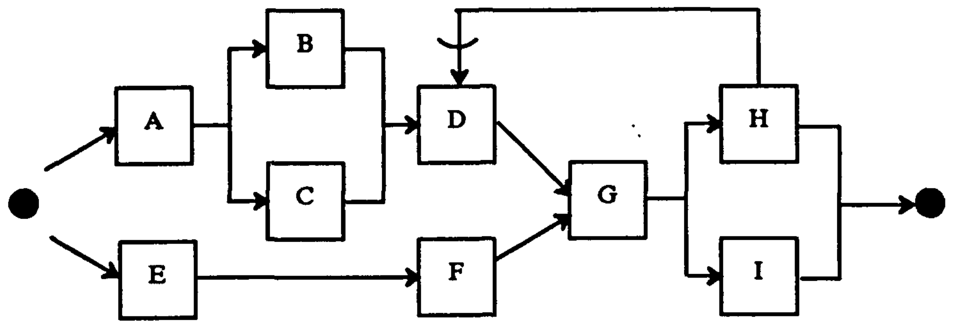


Figure 3-8. Example Process Behavior Model

Behavior relationships are used to show the ordering (full or partial) of activities. Figure 3-8 shows several ways in which activities are arranged. The primary arrangements of interest are:

- Origination: What happens first
- Sequence: When an activity is done, another may start
- Selection: When an activity is done, one of several others may start
- Iteration: When an activity is done, it may repeat or be part of a repeating series
- Dispatch: When an activity is done, several may start
- Rendezvous: When several activities are done, another may start
- Release: When any one of several activities are done, another may start
- Termination: What happens last

The following list provides details on interpreting Figure 3-8:

- ***Origination.*** Origination is shown as a small, dark circle with an arrow going out to one or more activities. In Figure 3-8, activities A and E are both originating activities. This process begins with both A and E executing in parallel.
- ***Sequence.*** Sequence, in its simple form, is one activity following another. In Figure 3-8, this is shown as activity F following E.
- ***Selection.*** Selection indicates that only one of the following activities may execute. For example, after activity A is done, either B or C may start, but not both. To indicate selection, always use a single departing arrow, then branch for each of the possible following activities.
- ***Iteration.*** Iteration is shown in Figure 3-8 by the arrow leaving activity H and pointing back to activity D. Note that this arrow has a small arc by the arrowhead pointing to D. This indicates an optional relationship. In other words, there might be a need to return to H from D, but not always.

- ***Dispatch.*** Dispatch is shown by multiple lines leaving an activity, or starting node, arriving at different activities. The originating node indicates this activity starts with the dispatch (parallel activation) of A and E.
- ***Rendezvous.*** Rendezvous is shown by several lines each separately arriving at an activity. In Figure 3-8, this is shown as activity G waiting on both activity D and activity F to finish. Note that this brings together the work of A and E.
- ***Release.*** Release is indicated by several lines joining into one line, and then that one line going to an activity. In Figure 3-8, when either activity B or C are done, activity D starts.
- ***Termination.*** Termination is shown by one or more arrows pointing to a small, dark circle. Consistent with the conventions just described, if several arrows each independently point to the terminal mark, it indicates that each of the activities must complete. If the arrows join before arriving at the terminal mark, then any one of the preceding activities serves as the final activity. In Figure 3-8, both H and I are potential terminal activities.

3.6.3 Architectural Relationships

Figure 3-9 shows an example of a process architectural model.

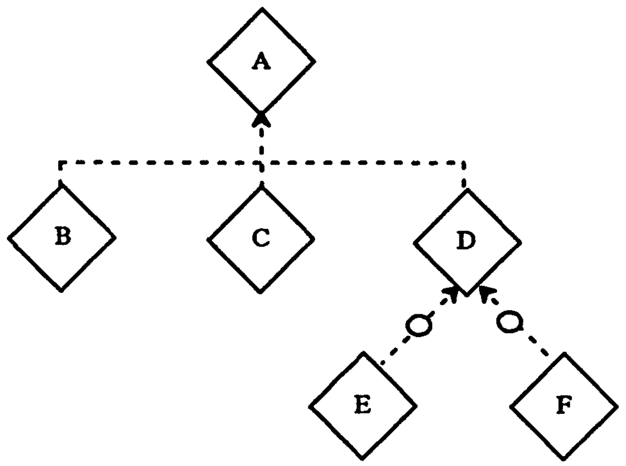


Figure 3-9. Example Process Architectural Model

Architectural relations show parent/child relationships. There are two primary types of relationships used in building the process model; these are:

- ***Composition.*** Composition relations indicate that the parent object is composed of, or built from, all of the child objects. Figure 3-9 indicates that role (or team) A, is made up of roles (or team members) B, C, and D.
- ***Specialization.*** Specialization relations show that the parent actually has one of several different forms. In Figure 3-9, role D is shown to be either role E or role F. As an example, role D may indicate the role of an inspector, and roles E and F indicate that there are two types of inspectors: key inspectors and regular inspectors. The circle by the arrowhead indicates a specialization relation. However, the circle can also be placed at the source end of the arrow. This is typically preferred if several arrows join into one.

3.6.4 Interface Relationships

Figure 3-10 shows a variety of interface relationships (represented by dashed lines) and is interpreted as follows.

- Activity A1 needs both role R1 and Role R2.
- Activity A1 always produces product P2, and sometimes ("optionally") produces product P1.
- Activity A2 needs role R2 and role R3.
- Activity 2 requires product P2, and may optionally use product P3.
- Product P1, if it is produced, is sent to some external "sink" (such as might be shown on, or at least implied by, a level-0 context diagram).

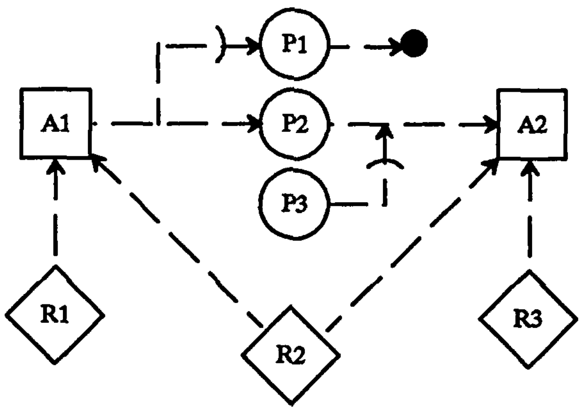


Figure 3-10. Example Process Interface Model

It is clear that this figure does not necessarily tell you everything you want to know. For instance, once the product P2 is available, can the Role R2 be shared between A1 and A2? Although this type of information can be found on or deduced from the templates, those details are lost in this particular diagram. However, adding more detail is not always the solution: cluttered diagrams quickly begin to confuse people.

With any diagram, you are deliberately eliminating details so that you may emphasize what you consider to be most important. With interface diagrams, the primary objective is to show, at an abstract levels, how roles, products, and activities all coexist.

3.6.5 Organizational Reporting Relationships

Figure 3-11 shows an example of a process reporting model.

Organizational relationships only exist between roles, and they exist to show, for any given role, who (or which roles) report to it, and to whom does this given role report. Figure 3-11 tells us that role D is reported to by roles A, B, and C. It also shows that role D must report to role F and, optionally, reports to roles E and G.

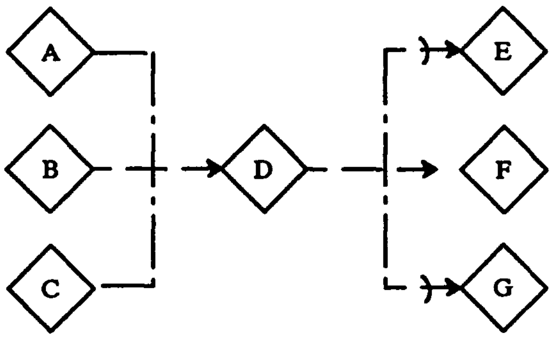


Figure 3-11. Example Process Reporting Model

3.6.6 Mixing Models

Because each of the roles has a different shape, and each of the relationships uses a different type of line, it is possible to build composite or overlay models that strive to show a variety of information and relations simultaneously. Generally, this should be avoided. However, if you want to combine two or more types of simple models, be sure to verify that the resulting hybrid actually improves communication and understanding. Remember, at Tier 1 the goal is always to facilitate accuracy, clarity, and communication. Cluttered diagrams quickly become a hindrance, and they are labor-intensive to maintain. The conventions presented in this section are specifically intended to help you build models with a minimum of effort, but which still communicate a significant amount of process information.

3.7. SUMMARY

Section 3 presented the conceptual foundation for MPDM; tiers of usage; template descriptions, at Tier 1, and their use in collecting, organizing, relating, and maintaining process information; and a set of graphical conventions that can be used to model your process.

In essence, Section 3 has described what the templates are. Section 4 describes how you use them.

4. FUNDAMENTALS OF DEFINING YOUR PROCESS

Congratulations—you have survived Section 3! Do not be concerned if you are feeling somewhat overwhelmed. You have just been through a lot of "mechanics" about templates and modeling, and it is too early to expect everything to make sense. As you advance through Sections 4 through 6, the overall picture will become progressively more clear. At this point in the guidebook, however, you should be comfortable with the following key concepts:

- Process definition is an information management problem, not a word processing problem.
- MPDM is a process definition methodology that relies extensively on information management principles.
- In MPDM, process data organization and management are achieved using a set of process templates.
- MPDM recognizes four tiers of process information, where each tier is progressively more formal and detailed in the information and relationships captured.
- Process guidebooks, training material, and similar products can all be generated using Tier 1 templates (the least formal tier).
- There are only three template classes used at Tier 1: activities, products, and roles.
- MPDM separates, yet integrates three primary types of process information: (1) information needed for your process model, (2) information needed for your model-derived process end-products, and (3) information that facilitates management, review, quality assurance, and configuration management of your evolving process information.
- There are nearly 30 other templates that extend MPDM, and none of them are important right now.

In Section 2 you received a high-level view of how to approach process definition. Section 3 provided details on the nature of and relationships between various types of process information. This section brings these two views together by providing step-by-step guidance on starting and conducting process definition, modeling, and end-product generation. Later, Section 5 will provide a complete example.

Section 4.1 and 4.2 provide information and guidance on MPDM techniques and template usage. Section 4.1 examines how to begin using templates to support developing process guidebooks. Section 4.2 extends this discussion by presenting a recommended scenario of detailed tasks for performing this work.

4.1 GETTING STARTED WITH TEMPLATES

As part of building a template-based model, you need to plan how you will make maximum use of the model when it is completed. Process templates and the corresponding graphical notation support the construction of process guidebooks directly from an electronic inventory of templates. If you are going to use this approach you should start building the model with an information management tool, such as a relational database, which will be used to produce the guidebook.

There are several advantages in using a database tool to maintain and update template-based process representations. This approach potentially removes the need to maintain the guidebook and its underlying model separately. When guidebooks can be derived directly from the templates, only the templates need to be maintained. Second, the templates can be automatically checked for consistency, completeness, conformance to structure rules, etc. The templates impose a natural organization to process information and provide explicit layers of abstraction for improved conceptual overview.

If you prefer to keep the information contained on the templates to a minimum, you can still use the templates, in an automated environment, to provide the skeleton of a process guidebook. Process analysts would then only need to annotate the information in the initial draft by adding supplemental material that expands upon what was directly extracted from the templates. Because the templates are tolerant of free-form descriptive text, information can all be reformatted and presented in guidebook format.

The following sequence of activities is typical of an organization's ongoing efforts to develop and improve process guidebooks using the template-based approach:

1. Analyze the existing process by building high-level graphical and template-based models that identify the principal activities performed and the principal products involved in the process.
2. Extend the models by adding additional process details, including roles, and verify the model with process experts and process users.
3. Use the models as a foundation for producing a draft guidebook.
4. Modify the model based on insights gained from expert and user reviews of the draft guidebook.
5. Increase the detail and precision of the model.
6. Update the guidebook to reflect the latest model.
7. Use the guidebook on a pilot project.
8. Analyze project metrics.
9. Evaluate potential process improvements.
10. Construct revised graphical and template-based models reflecting approved changes.
11. Go to Step 3.

These steps can result in defining an increasing set of reusable process assets that you can use to support multiple projects.

It is important not to confuse improving the model with improving the process it is representing. Improvements to a model are always done from the perspective of the process implications of changing some given activity, especially with respect to its impact on other activities. Often, changes that appear easy to make on the templates are actually quite hard to carry out within the real process. Conversely, what appears difficult or awkward to change within a model (graphical or templates) may actually be very simple to change in the process itself.

In all cases, the process engineer's orientation must be on improving the process. The model is only a facilitation tool: a means toward an objective. The process engineer should strive to constrain proposed or improved process models to those that are realistic within an organization. From the perspective of support for process improvement, the goal is not to develop the ideal model; the goal is a model that gives insights into the ways that **feasible** incremental process improvement can be achieved. It is important to remember that there are distinct differences between improving the clarity of a template model, improving the usefulness of the templates themselves, and improving the process represented by the templates.

To summarize, the key importance in the use and optimization of templates is the principle that defining a process model proceeds one stage, or cycle, at a time. Each stage should be comparatively small and relatively low risk. Each stage should be evaluated in terms of the value derived from process definition in relation to the effort required to construct that definition. This technique corresponds directly to the "goal-driven" approach advocated in Section 2.

4.2 MPDM TASKS

This section describes the MPDM tasks for performing process definition and modeling. The approach reflects the sequence of activities presented in Section 4.1. This is a recommended approach, but by no means the only approach. It can serve as a good point of departure for an organization that does not have a history of, or strong preference for, using a different technique. It uses only the templates classified as Tier 1 in Section 3.

The scenario uses a top-down approach to process definition based upon decomposing the process into tasks and products to increasing levels of detail. The decomposition process is very similar to that recommended in other process modeling techniques such as IDEF0 and SADT. If a large number of reusable process kernels should exist in an organization, a scenario based upon process composition beginning with the process kernels could be preferable. Another approach would be to start with the products that are the output of the process and to identify activities that create or transform intermediate products to produce the process outputs. The scenario provided in Figure 4-1 is a practical approach to process definition and can be used in a wide variety of situations.

There are numerous ways that you can use the templates to perform process definition and modeling. It is highly recommended that regardless of the approach selected, there should be some element of cyclic refinement. A relatively low-risk, cost-effective approach to building process models is to initially build a complete—if abstract and simplified—representation of the process. When this has stabilized, increasingly detailed information can be captured in the templates. Many process engineers prefer to use the graphic notation to capture the high-level representation of the process. This approach enables a greater degree of detail to be gradually introduced and a progressively more explicit and complex model to be constructed. The overall model is then cyclically extended to include more information in both the graphical and template-based representations.

As shown in Figure 4-1, the template development scenario (using Tier 1 templates and fields) consists of eight major tasks:

1. Identify the external interfaces to the process.
2. Gather information to define the major activities composing the process and the related products.
3. Build a graphical model based upon the activities and products identified.
4. Define activities and relationships using the templates.
5. Define products using the templates and extend the graphical model to be consistent.
6. Verify the completed model with process experts and revise the graphical model and the templates as required.
7. Define the roles involved in the activities and extend the graphical model to show these roles.
8. Produce draft guidebook (Tier 1) for review by process experts and potential users.

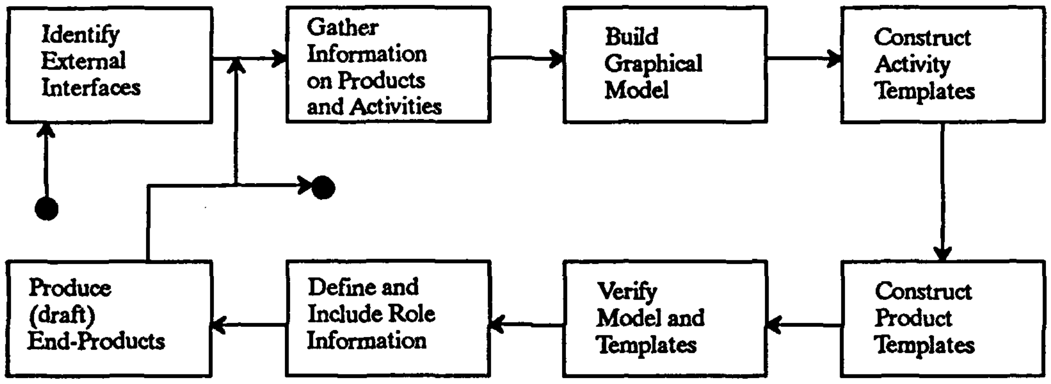

Identify External Interfaces
Gather Information on Products and Activities
Build Graphical Model
Construct Activity Templates
Construct Product Templates
Verify Model and Templates
Define and Include Role Information
Produce (draft) End-Products

Figure 4-1. MPDM Tasks

Remember that this continues as a cyclic process that regularly verifies accuracy, clarity, and ability to generate the desired process end-products. As related in Section 4.1, upon achieving sufficient quality in your process definition, you then pilot the definition on one or more projects, update as a result of the pilot, and then release to the intended users.

4.2.1 Task 1: Identify External Interfaces

To define the scope of the process definition to be created, you must decide what part of the software life cycle is to be represented. A good way to start is to develop a list of all the products created by the process to be defined (i.e., all process outputs) and all of the products needed to perform the process (i.e., all process inputs). The process is operationally defined as the sum of all the activities required to produce the output products from the input products. Figure 4-2 shows an example. Note that at this level the process to be defined is always represented as a single "activity." Decomposition of process is deferred until after you have defined the required external input and external output products.

4.2.2 Task 2: Gather Information on Products and Activities

Task 2.1: Define Major Activities

The first step of Task 2 is to define the major activities believed to compose the process. Unless you are already familiar with the process, you will need to gather information from existing documentation

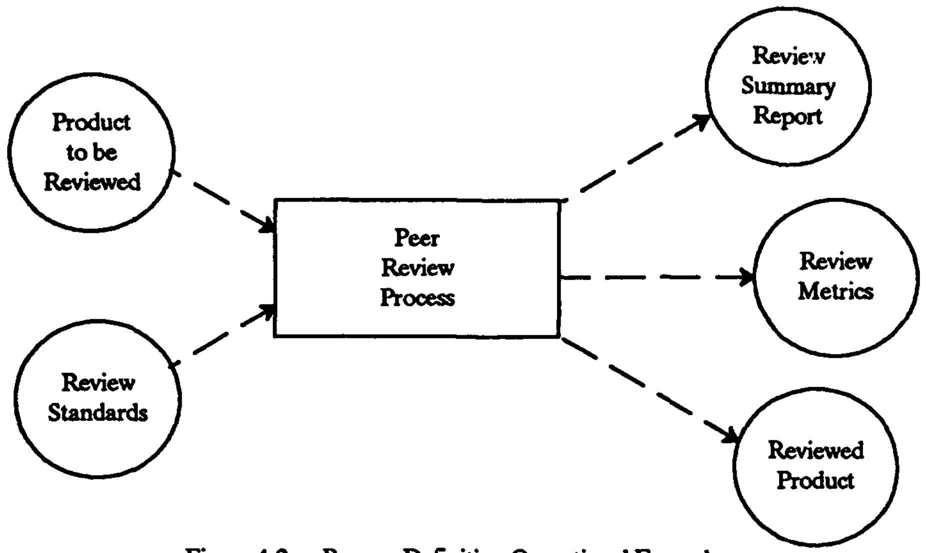


Figure 4-2. Process Definition Operational Example

and from interviews, as discussed in Section 2. The goal is to identify three to ten activities that, in sum, represent the process. All aspects of the process must fit into one of the activities, and all of the external products must be produced by one or more of the activities.

This initial decomposition of the process into a small number of activities provides a structure for organizing the information you gather. The names of the activities identified are likely to change as you review your work with persons knowledgeable about the process and the terminology used in the area. You should try to initially limit the process decomposition to six major activities, because the template-based model you are building will be much easier to understand if you can organize the process description into a few major activities with clearly defined relationships between them.

Once the names of the major activities have become relatively stable, the next step during information gathering is to build an indented list of activities. The purpose of the indentation is to reflect how higher-level activities can be broken down into lower-level activities or subactivities. For each item on this list, provide a unique identifier and a brief explanation of the activity's purpose and description.

At this stage, it is sufficient to attempt to define only the first two or three levels of the activity breakdown. The first, or highest, level defines the major activities occurring within the process. The second level defines the subactivities that make up each high-level activity. For example, subactivities of a project management activity might include:

- Perform cost/benefit analysis.
- Evaluate alternative solutions.
- Construct PERT chart.

At the third level, list the primary tasks that make up each activity. "Tasks" denote an activity that, at this time, is not further decomposed into subactivities. For example, the "evaluate alternative solutions" subactivity might further decompose into three tasks:

- Evaluate commercial solutions.

- Evaluate alternative in-house solutions.
- Evaluate consequences of doing nothing.

Of course, you may also decide that no further decomposition is necessary. For instance, you might decide to treat "Perform cost/benefit analysis" and "Construct PERT chart" as tasks.

Task 2.2: Define Internal Products

The second step of Task 2 is to define the internal products associated with the previously identified activities. An internal product is a product that is produced by one activity and used by another activity toward producing an external product (or an output of the process). Internal products are frequently produced from external products that are inputs to the process.

Products can be described in terms of levels of composition, and an indented product list makes it easy to show the levels. For example, if an output is "Software Product," an example of an indented list is:

```
Software Product
    Software Source
    Software Object
    Technical Documentation
        Administrator Guide
        User Guide
```

The indentation shows that the Software Product is composed of three major items: Software Source, Software Object, and Technical Documentation. Technical Documentation is composed of two guides. Initially you can use abstract names, such as "Software Product" for collections of related products and later define lower levels of composition as more detailed templates are created. Next, for each *item on the list, provide a unique identifier and a brief explanation of the product's purpose and description.*

As you define the internal products, you may need to update the list of external products, and the context diagram if you created one. Updating is needed when a new external input is required so that an activity can produce an internal product, which eventually contributes to an external output.

After completing this step of Task 2, you have identified the external products, the major activities composing the process, and the internal products used and produced by the activities. As information gathering continues, and others review your work, you should expect iterations in the names of the activities and products and in the composition of the indented lists.

4.2.3 Task 3: Build Graphical Model

To facilitate organization of the evolving process description and its review by others, you will need to create a high-level graphical model of your initial results. Using the indented list of activities and products as a reference, construct a high-level model based upon the graphical notation discussed in Section 3.6. This particular model will contain two types of objects: activities and products. These objects will be combined using two types of relations: architectural (to show decomposition of activities and products) and behavioral (to show that activities have an order of occurrence).

The indented lists provide the information needed to create the architectural relationships. However, since lower levels of the indented list structure are likely to change as you gather more information, you should restrict the initial graphical model to one level higher than the lowest level of decomposition in the indented list. As you become more confident of the indented list decomposition, you can add additional detail to the graphical model.

4.2.4 TASK 4: CONSTRUCT ACTIVITY TEMPLATES

This task consists of several steps which are all organized around filling in the Tier 1 fields in an activity template. If you have not created an indented list of activities to three levels that you feel is reasonably stable, you should perform enough information gathering to do so before beginning this task; you will avoid creating and later deleting templates as the activity decomposition is revised in major ways.

Task 4.1: Establish a Template for Each Activity on the Indented List

The first step of Task 4 is to establish a template for each activity on the indented list and fill in the basic information in the template header. If you are using an information management tool, some of the information, such as the Version Number and Date fields, may be inserted automatically. Because each template needs a unique identifier, use the suggestions in Section 3 to develop a useful, consistent, and intuitive naming convention for constructing unique identifiers for each activity. Fill in any other fields with the information already known (e.g., the Description or Purpose fields).

Task 4.2: Identify the Activity Hierarchy

The second step of Task 4 is to identify the activity hierarchy. Each template has a field for noting whether an activity has been decomposed on the indented list into subactivities or tasks (i.e., has child activities). All subactivities, by definition, have one or more parent activities. Based upon the activity hierarchy represented in the indented list, fill in, as required, the child activities fields and the parent activities fields. Detail the two-way relationships for all activities, unless your information management system requires you to only specify one direction (while it derives the other direction). You can also use the architectural models developed in Task 3 as aids in defining parent and child relationships on the templates.

Task 4.3: Define All Products

The third step of Task 4 is to define all products generated or used by the activity. Higher level activities will typically reference higher level products, and tasks will reference lower level or final products. If meaningful, also indicate whether a given product, within a given activity, is required or optional. The graphical model can help you confirm that you have captured all the relationships. Conversely, if you discover additional relationships during this task, you should update the graphical model as appropriate.

Task 4.4: Define Invariants

The fourth step of Task 4 is to define invariants. Invariants are anything that must remain in effect for the entire duration of the activity. These are entry criteria that must always be "true." If there are any special invariants for the activity you are describing, be sure to identify and explain them.

Task 4.5: Define Entry Criteria

The fifth step of Task 4 is to define entry criteria. For each activity define the entry criteria, unless the default entry criteria (the availability of each of the specified input products) is applicable. If only a subset of the input products is required for the activity to begin, the entry criteria should define the subset. The entry criteria can be defined in natural language, making reference to information such as whether other activities have been completed, are underway, etc.

The least error-prone approach is to start with the high-level activities, since the entry criteria for a subactivity is usually a subset of the entry criteria of its parent. As the entry criteria are defined, ensure that they are consistent with the implications of the graphical model built in Task 3. If need be, update or alter the graphical model as insights are gained from ongoing analysis and the effort of filling in the templates.

Task 4.6: Describe Internal Process

The sixth step of Task 4 is to describe the internal process. An activity's internal process is the principal description of the work that characterizes it. This description is done in terms appropriate to abstraction level of the activity. Generally, you want to describe what the activity is and/or how it is performed.

There are numerous examples of internal process descriptions on the templates shown in Section 5 and in Appendix A. Generally, at progressively lower levels of detail, you will include proportionately more "how-to" information. At higher levels, the internal process will typically read more like a policy statement, as opposed to having procedural detail.

The states of products (i.e., "once the design document has been inspected...") are frequently used in defining internal processes. Task 5.4 describes how to define this type of state information.

If there are constraints on how the internal process is done, describe them, for now, in natural language. An example of such a constraint is the need to follow a certain government standard in performing the activity. It may be useful to describe these constraints later using constraint templates (a Tier 2 template described in Volume 2 of this guidebook).

Task 4.7: Define Exit Criteria

The seventh step of Task 4 is to define exit criteria. For each activity, define the exit criteria, unless the default exit criteria (the production of each of the specified output products) is applicable. Other criteria may need to be met for the activity to be complete (i.e., a quality assurance [QA] review has also been satisfactorily completed). The exit criteria can be defined in natural language, making reference to information such as whether other activities have been completed, etc.

At this point, you have described the activities in detail. The next step is to produce an equivalent level of detail in the product templates.

4.2.5 Task 5: Construct Product Templates

This task consists of several steps which are all organized around filling in the Tier 1 fields in a product template.

Task 5.1: Establish a Template for Each Product on the Indented List

The first step of Task 5 is to create a template for each product on the indented list and fill in the basic information in the template header in a manner similar to what you did for the activity templates.

Task 5.2: Identify Product Hierarchy

The second step of Task 5 is to identify product hierarchy. There are spaces on the templates for noting parent/child relationships between high-level products and the subproducts of which they are composed. These relationships will closely follow the indentation levels in the product indented list. The graphical model is also a useful reference (specifically, product architectural relationships—including both inclusion and specialization). Use the Parent Templates and Child Templates fields to document these relationships to whatever depth is necessary.

Task 5.3: For Each Product, List All Contributing Activities

The third step of Task 5 is to list all contributing activities for each product. This step is likely to be done automatically if you are using a database system to support development of the templates. If not, once all the activity templates have been updated to reference applicable products, each of the product templates needs to be updated to reference the applicable activities (i.e., the Activity field on the product template). In this way, a cross-reference exists to facilitate verifying the integrity of the process definition. It is also useful to specify, on the activity template, whether the product is required or optional. The graphical model can help you confirm that you have documented all the relationships. Of course, update the graphical model as appropriate.

Task 5.4: Identify Product-Specific States

The fourth step of Task 5 is to identify product-specific states. State information, which was discussed briefly in Section 3, is most frequently used to describe products. For example, a design document might have states such as "being drafted," "in review," "awaiting design inspection," "being updated," "completed," "approved by QA," etc. The state information can be used to make the internal processing descriptions of activities more precise, as well as being used in formulating entry and exit criteria.

You can define state names specific to a product or use a default set of names. Product states presented in Section 3 include:

- Planned
- Scheduled
- Authorized
- Enabled
- Under Development
    a. Phase 1 Completed
    b. Phase 2 Completed

c. Phase N Completed

- Ready for Review
- Reserved for Review
- Under Review
- Review Completed
- In Rework
- Completed
- In Suspension
- Cancelled
- Approved
- Released
- Fielded
- Retired

These are examples of how you can distinguish different states for different objects. Generally, strive for the fewest number of states needed to support the model. For each product template, only add those states that are necessary or useful for defining relationships between the product and activities.

Task 5.5: Define Product State Transitions

The fifth step of Task 5 is to define product state transitions. If you desire increased detail, then for each product, define how it advances through various product states as a function of the activities that govern the evolution of that product. This additional information will give you a different perspective on the process and may cause you to define new activities or update the definitions of activities.

Task 5.6: Update Internal Process to Include Product State References

The sixth step of Task 5 is to update the internal process to include product state references. Now that you have explicitly defined products and their corresponding states, you can upgrade the internal process within an activity to include references to the states of products. This can be especially useful for representing the review or quality assurance phases of a particular activity.

Task 5.7: Update Entry and Exit Criteria to Include Product State References

The seventh step of Task 5 is to update entry and exit criteria to include product state references. Similar to the previous step, for all activity templates, you can upgrade the entry and exit criteria to include references to the states of products that are either expected by or released by a particular activity.

When you have completely defined the products, update the graphical model as required to be consistent with the templates. The graphical model will be the principal vehicle for verifying the completed model or models with process experts and other reviewers.

4.2.6 Task 6: Verify Model and Templates

At this point, you have a process model that describes both activities and products, although it may not yet contain the detail necessary to produce a guidebook for carrying out the process. However, before proceeding further, you should verify the correctness and consistency of this preliminary model to the maximum extent possible.

One possible verification vehicle is to conduct model walkthrough for reviewers knowledgeable about the process. In the walkthrough, the model developer leads the reviewers through each step of the process, answering questions and identifying errors and missing information as the walkthrough proceeds. The graphical model, backed up by the templates, is the basis for the walkthrough.

4.2.7 Task 7: Define and Include Role Information

After you update the graphical model and the templates based upon the review results, you can then extend the model further to include the personnel and organizations that perform activities. This is the last stage of extending the scope of information in your model. Afterwards, each successive iteration is primarily used to capture detail and improve usability.

The definition of roles that perform activities proceeds in much the same manner as the identification of activities and products. First, develop an indented list of roles based upon the information previously gathered and assumptions about who is likely to perform the activities previously identified. Subsequent steps involve updating the graphical model to include supports and establishing support templates for the roles on the indented lists.

Task 7.1: Build an Indented List of Supports

The first step of Task 7 is to build an indented list of supports. This list should detail the roles needed to support the work domain. The term "roles" is used in the broadest sense; therefore, not only do individuals perform roles but so do teams, entire divisions, etc.

Task 7.2: Extend the Graphical Model to Include Supports

The second step of Task 7 is to extend the graphical model to include supports. Using the support or role indented list as a reference, extend the graphical model using the notation discussed at the end of Section 3. Your new graphical model will now contain three types of objects: activities, products, and roles. It will likely contain four types of relations:

- Architectural—to show decomposition of activities, products, and roles
- Behavioral—to show ordering of activities
- Interface—to bind activities and products and to bind activities and roles
- Reporting—to show who reports to whom

Task 7.3: Establish a Template for Each Role on the Indented List

The third step of Task 7 is to establish a template for each role on the indented list. As in the prior tasks, when establishing the template, attempt to fill in any information already known.

Task 7.4: Identify Support Abstractions

The fourth step of Task 7 is to identify support abstractions. These abstractions are two-way relationships and, as with the other classes of templates, these are architectural or parent/child relationships. Keep in mind that architectural relations are of two types: composition and specialization.

An example of levels of abstractions within roles is:

- Programming Team
  - Team Leader
  - Programmer (3 to 6)
  - Technical Support (1 to 2)
- Line Manager
- Administrative Manager
- V&V Department
  - Statistical Group
    - Statistical Group Manager
      - *Senior Statistical Group Manager
      - *Junior Statistical Group Manager
    - Statistical Group Engineer
  - Dynamic Test Group

The indentation on the role indented list and the architectural graphical model can both be used for initial guidance on documenting parent/child relationships on the role templates.

Additionally, as shown on the example, it improves clarity on the indented list if you show when the children of a parent exist as a special case of that parent. For example, the Statistical Group Manager role in the indented list is indicated as really being of two different types: Senior Manager or Junior Manager. Using a convention such as a leading asterisk to clearly distinguish the type of architectural relation will greatly facilitate verifying the integrity between the indented lists, the graphical models, and the templates.

Task 7.5: For Each Activity, List All Needed Roles

The fifth step of Task 7 is to list all needed roles for each activity. If appropriate and desirable, also note whether a support is needed exclusively by an activity, whether it can be shared, and whether that support can be considered optional. For example, secretarial support might be needed only in a shared capacity.

Task 7.6: For Each Role, List All Supported Activities

The sixth step of Task 7 is to list all supported activities for each role. This is the cross-referencing step and may be done automatically if you are using a relational database system. It details the companion relationship between supports and activities, whereas the prior step defined the relationship between activities and supports. If some roles identified during information gathering appear to support no activities, this is a good indication that some activities may have been omitted from the model.

4.2.8 Task 8: Produce Draft End-Products

The steps in this task outline how to start generating guidebook material (Tier 1) from your process model.

Task 8.1: Develop Navigation and Extract Algorithm

The first step in Task 8 is to develop navigation and extract algorithm. To produce the end-products you want, you have to answer two key questions:

- In what order should the templates be navigated?
- For each template encountered, what fields need to be exported?

Section 5 provides examples of ways you can answer these two questions. Briefly, the steps you take are:

1. Select a primary "view" or orientation for your end-product.

   For guidebooks, this is most commonly an activity-based view; however, it is also highly useful to generate role-based guidebooks. This view will be heavily based on the audience for the end-product.

2. Define how you will navigate the database to find and select templates in accordance with your primary view.

   By far, the most common way to do this is through architectural relations. For example, if you selected an activity-based view, then you might navigate the templates as follows. Find the root template, export relevant information, and see if it has any children. If so, find the first child template and export its relevant information. If that template has its own children then drop another level and export that information. When all the child templates under a parent have been visited, then return to the parent and look for additional siblings at that level. Continue until you have returned to the root node and no further (unvisited) child templates exist.

3. Select the auxiliary information you want and when you want to export it.

   For example, maybe you want to show both the products and the roles related to each of the activities. If so, each time you finish exporting information from an activity template (while navigating the activity architecture), you may want to visit each of the associated product templates and export product information, such as Name, Brief Description, Usage, and Impact. When done with all the products associated with a given activity template, you may want to similarly visit each associated role template and export one or more fields from there. Again, this logic would repeat for each activity template visited as you navigated the architecture.

4. Import and "polish" using an appropriate end-product tool.

   It is not worth trying to turn your database into a word processor. Once you have exported the information from the database, you should load it into an end-product tool to do any remaining work. For instance, if you intend to create guidebooks, then use a word processor for final cleanup. Take advantage of any additional options the word processor provides. For example, many word processing packages have facilities for automatically building a table of contents.

Task 8.2: Add End-Product Information

The second step in Task 8 is to add end-product information. When you first look at the raw export information from your process model, it will likely read nothing like a guidebook. This is because virtually all documented processes include not only process information, as in Type-A fields (see Section 3) but also transitional or presentation related information that helps keep the reader oriented. Your goal now is to add additional information to the templates that is specifically intended to improve the *presentation* and flow of a particular process end-product.

This step proceeds as follows. Start working your way through the end-product evaluating it for consistency, completeness, depth, scope, and flow. If you encounter problems, you have to decide whether the problem is in one of the following areas:

- The fundamental process model
- The process information contained within the model
- The supplemental (or Type-B) information
- The way in which the model was navigated
- The fields that were selected to export

Typically, you will quickly stabilize on the navigation and extract algorithm. Next, you will correct or otherwise improve the basic process model information. Then, the only remaining work is to continue to add or improve the Transitional Text fields until you achieve a flow of information that progresses smoothly.

Task 8.3: Review of End-Product

The third step in Task 8 is to review the end-product. Once the process model (including end-product specific, Type-B information) is reasonably complete, you should produce another draft guidebook for review by both process experts and typical technical staff members who will actually use the guidebook.

Again, remember that as you strive to improve the quality and usability of your process end-products, you will do so by continuing to improve:

- The fundamental model
- The information contained within that model
- The ways in which you navigate the model
- The information that you extract from the model
- The "packaging" you put around and between that information to augment specific end-products

In other words, you have many alternatives for improving quality.

Section 5 shows you a complete example that includes indented lists, graphical models, filled-in templates, extract algorithms, and an example end-product.

5. EXAMPLES: PEER REVIEW PROCESS

Section 5 shows examples of the material discussed in Section 4. Consistent with the general advice in this guidebook, the process being defined and modeled is a "process kernel." From this perspective, it is a standalone or encapsulated process that is "plug-compatible" into a variety of different life-cycle models.

The process kernel in these examples is the Peer Review process. The example elaborates a review process that is loosely based on the Fagan Formal Inspection process (Fagan 1986). The primary objective of this section is to show you a reasonably complete example of how to use MPDM to define a process and generate process end-products. This section is **not** intended to detail an actual review process. The key objective is to show you, as simply as possible, key aspects of MPDM.

Additionally, for space consideration, only a few templates are shown here. Appendix A, Volume 2, contains the complete set.

5.1 LEVEL-0 CONTEXT DIAGRAM

As stated in Section 4, Task 1 in MPDM is to construct a Level-0 context diagram that details how the process you are defining relates to products needed from and delivered to the external world. Figure 5-1 is a simple example of the information needed and produced by the Peer Review process.

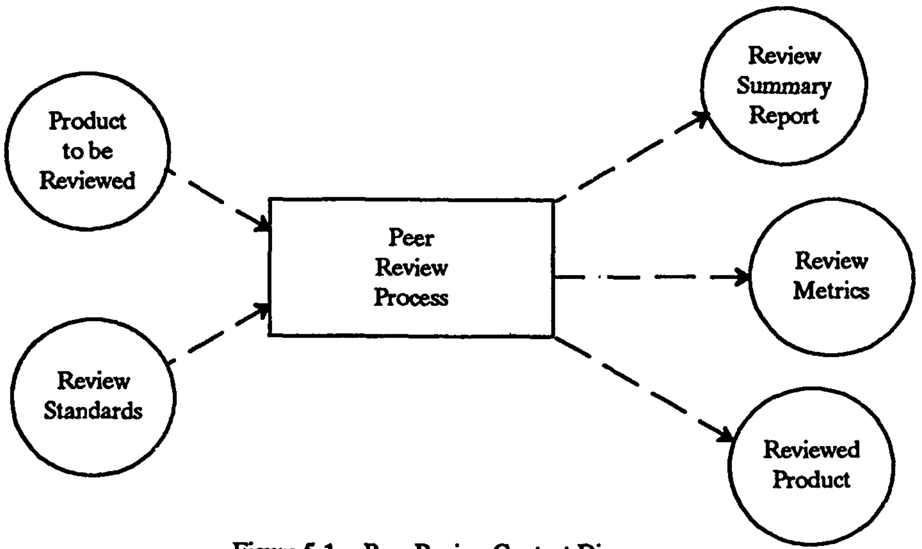


Figure 5-1. Peer Review Context Diagram

5.2 INDENTED LISTS

The next step, Task 2, is to build indented lists of the major activities (Task 2.1) and the internal products (Task 2.2).

For this example, the major activities are:

Peer Review Process

- Inspection Activities
    - Planning
    - Overview
    - Preparation
    - Inspection Meeting
        - Conduct Review
        - Decide Upon Reinspection
    - Rework
    - Follow-up
- Causal Analysis

The major products are:

Peer Review Products

- Product to Be Inspected
- Review Memos
    - *Invitation
    - *Results
- Review Metrics

Note that this indented list follows the convention of indicating "specialization" children with a leading asterisk.

Additionally, Task 7 of MPDM involves extending your process model to include information on roles. This process is quite similar to that followed for activities and products. The major roles in this example are:

Peer Review Roles

- Review Team
    - Moderator
    - Scribe
    - Reader
    - Inspectors (are...)
        - *Key Inspector
        - *Regular Inspector
    - Developer
- Review Coordinator

5.3 GRAPHICAL MODELS

Task 3 involves building one or more initial graphical models to help you and others understand the primary activities and product objects and the principal relationships that exist between them. Figures 5-2 and 5-4 are examples of architectural models for activities and products. Figure 5-3 is behavioral model for the activities. Figures 5-5 and 5-6 two interface models: one at a high level of abstraction and one at a more detailed level.

Figure 5-2 shows an architectural model of the Peer Revie\ ocess. The Peer Review process is composed of two principle activities: Inspection activity and Causal Analysis activity. Inspection activity is composed of Planning, an optional Overview activity, Preparation, Inspection Meeting, an optional Rework activity, and Follow-up. The Inspection Meeting activity consists of two principal parts: a Review activity, and a Decide activity where the inspectors determine whether a reinspection is necessary.

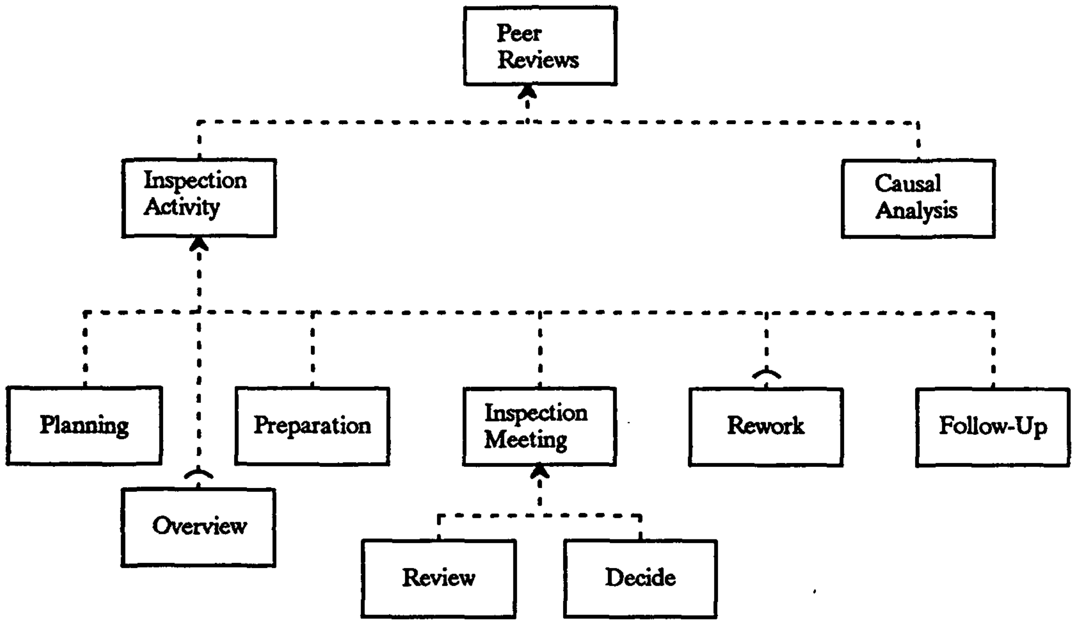


Figure 5-2. Activity Architectural Model

Figure 5-3 shows a behavioral model of the Inspection activity. This model shows that the review process starts with Planning, then either the Overview occurs, or the process may proceed straight into Preparation. After Preparation, the Inspection Meeting occurs. Following the Inspection Meeting, the Peer Review process may be complete, or there may be a need for Rework. If the latter, then after the Rework is done, a Follow-Up activity occurs. The Follow-up activity may be the last activity of the Review process, or if a reinspection is necessary, then the process loops back to the Preparation activity.

Figure 5-4 shows an architectural view of the products involved in the Peer Review process. The three major products are: Product to Be Inspected, Memos, and Metrics. Also shown is the fact that there are two types of memoranda: Invitation Memo and Results Memo.

Figure 5-5 shows an interface model of the high-level activities and products and resources that are involved in the Peer Review process. This model shows that Inspection activity needs the Product to Be Inspected, uses (or generates) Memos, and requires a Review Team. The Inspection activity generates a Metric product, which in turn is used by the Causal Analysis activity (which also requires a Review Team).

Figure 5-6 shows a slightly more detailed interface mouel of activities and products in the Peer Review process. Note that a detailed model does not necessarily show information seen at the higher level of

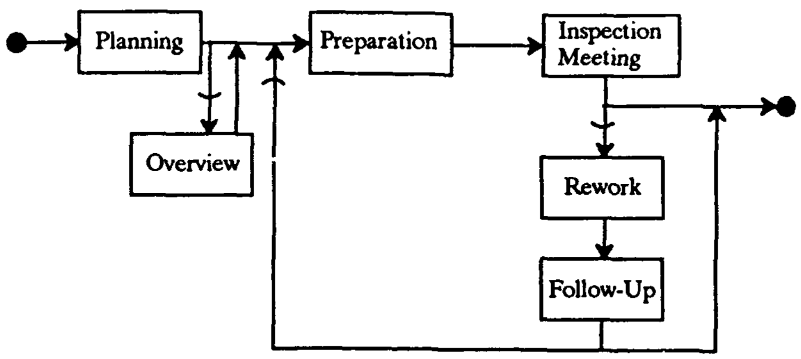


Figure 5-3. Activity Behavioral Model

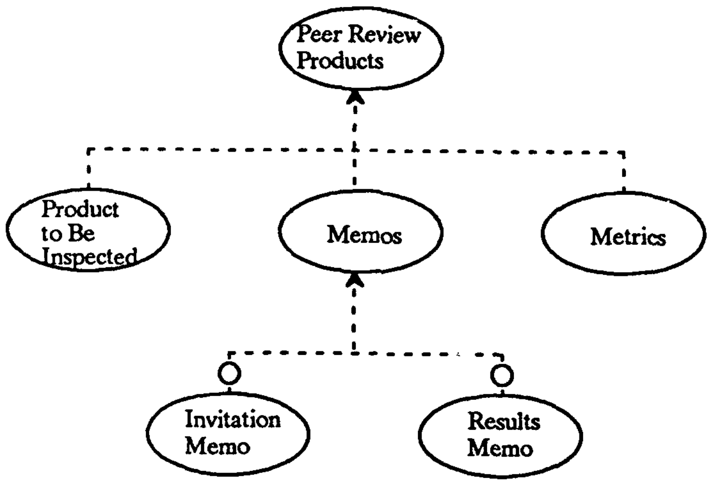


Figure 5-4. Product Architectural Model

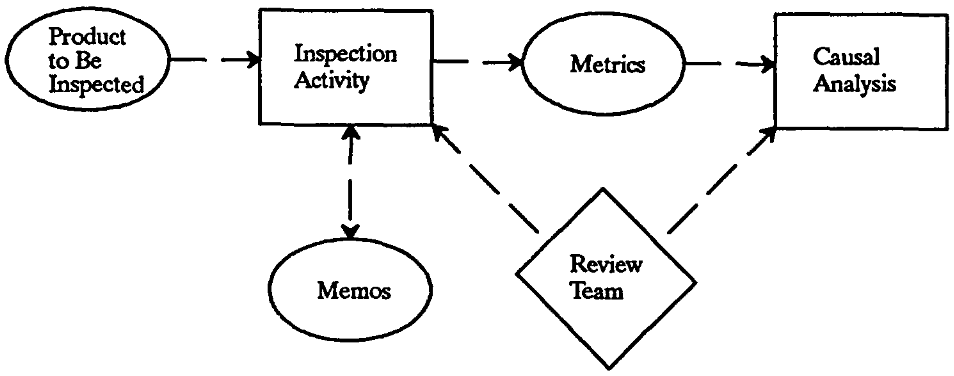


Figure 5-5. High-Level Interface Model

abstraction. Typically, the two views shown simultaneously create too much clutter. You should select the desired level of abstraction and then concentrate on showing the objects and relationships that exist at that level. This model shows that the Invitation Memo is generated by the Planning activity and used by the Overview, Preparation, and Inspection Meeting activities. This model also shows that the Product to Be Inspected originates from the external world, is needed in each of the activities, but is returned to the external world. Both the Inspection Meeting and the Follow-Up activities contribute to Metrics and the Results Memo.

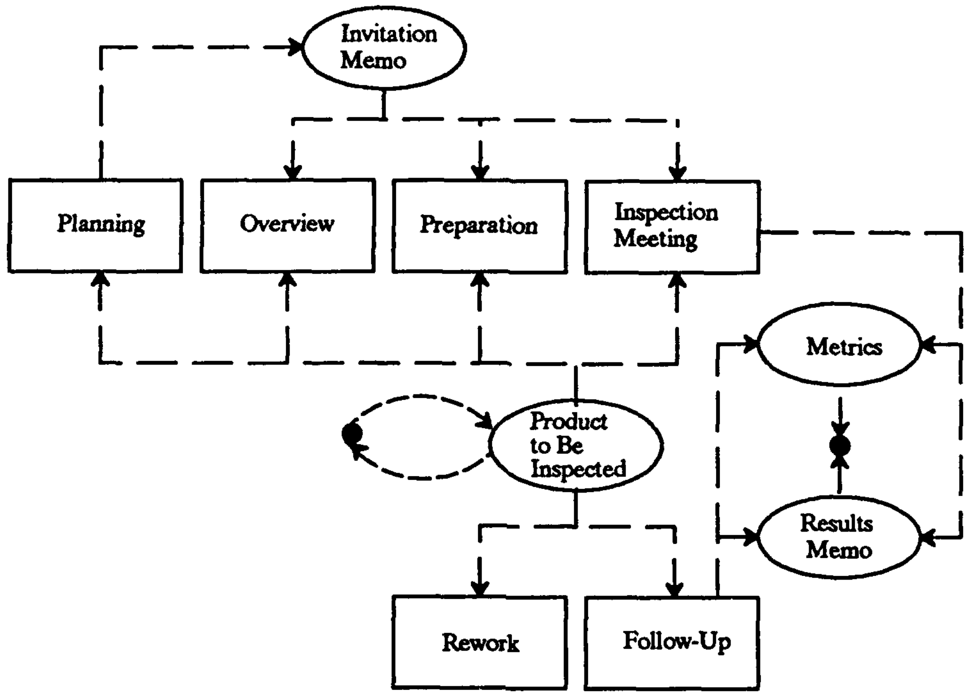


Figure 5-6. Detailed Interface Model

Again, these are simple diagrams and many details are not shown. However, even diagrams like this can help you quickly communicate to your audience the primary process objects in your model and how those objects relate to each other.

5.4 TEMPLATES

The full set of templates is available in Appendix A, Volume 2. However, a couple examples of each of the templates are shown Section 5.4.1 shows activity templates, Section 5.4.2 shows product templates, and Section 5.4.3 shows role templates.

5.4.1 Peer Review Activity Templates

Task 4 of MPDM involves capturing additional information about your activities by using the templates.

Table 5-1 shows the example activity templates in this section.

Table 5-1. Example Activity Templates

Activity Name	Unique Identifier
Peer Reviews	INS
Inspection Planning	INS_PLAN
Inspection Meeting	INS_MTG
Inspection Meeting Review	INS_MTG_REV

5.4.1.1 Activity Template: Peer Reviews

5.4.1.1.1 Name

Peer Reviews

5.4.1.1.2 Unique ID

INS

5.4.1.1.3 Brief Description

The peer review activity is based upon the formal inspection process.

5.4.1.1.4 Overview Description

The inspection method is one technique for the static verification of an artifact. Developed by Michael E. Fagan of IBM (Fagan 1984), the primary objective of this method is to detect defects in an artifact in an efficient and effective manner. Frequently referred to as a "formal inspection," you can use this method to verify any artifact generated during the development process, although considerable focus has been given to applying this method to preliminary design, detailed design, and code artifacts.

The inspection method consists of well-defined roles and activity stages. Associated with the roles are specific responsibilities. Additionally, entrance and exit criteria and objectives for each activity stage and mechanisms for recording and reporting defects ensure consistent application of this method.

The efficiency of this approach depends on focusing the participants on the overall objective of the method, which is defect detection. The effectiveness of this method depends on proper training of the participants and commitment by management to adhere to the prescribed process.

As stated earlier, you can use the inspection method to verify any artifact generated during the development process. This section, however, states the specific purpose for preliminary design, detailed design, and code inspections.

During the preliminary design phase, you hold preliminary design inspections to inspect the design documentation to verify that the requirements are satisfied and that the product will be maintainable, adaptable, and of high quality.

During the detailed design phase, you hold detailed design inspections to inspect the design documentation to verify that the design has been refined correctly, is defined to a level that allows coding to begin, and satisfies assigned product requirements. You inspect the design to ensure that it is maintainable, adaptable, and of high quality.

During the implementation phase, you hold code inspections to inspect the code to verify that it implements the detailed design correctly and completely.

The project team must baseline all reference material (i.e., specifications, requirements, documents) prior to the inspection meeting. For example, a preliminary design inspection cannot be held until they have baselined the requirements. An example of a requirements specification document is a system requirements specification (SRS).

5.4.1.1.5 Summary Description

It should also be noted that this method supports two types of tailoring. The first includes tailoring of guidelines and checklists to create an inspection process that adheres to the method and is relevant to the organization. You must do this type of tailoring to ensure a valid inspection process. The second type of tailoring is to modify the inspection method. This type of tailoring is optional. Additionally, tailoring of the method is subject to the limitations described below. The following discussion presents the mandatory tailoring followed by the optional tailoring.

Use of the inspection method requires that your organization establishes supporting mechanisms. These supporting mechanisms include entry criteria checklists, preparation checklists, and reinspection criteria for each artifact type that will be inspected. While there are many examples of these, you must tailor them to your organization's goals and the specific artifact being inspected.

Your organization may also tailor the classification of the defects. Typically, you classify defects, at a minimum, by type and severity. You may also classify each defect by category (i.e., missing, wrong, extra).

Because the objective of the inspection method is to find defects early in an efficient and effective manner, the focus is clearly on **defect detection**. One approach to tailoring the method, therefore, is to remove the causal analysis stage. The results of the causal analysis activity are intended to improve the development process to prevent the defect from occurring in the future. This stage emphasizes **defect prevention**. For organizations introducing the inspection process, the initial focus is probably on defect detection. The causal analysis activity could then be added in the future.

A second type of optional tailoring is to combine multiple inspections into a single inspection. This situation seems more frequent when inspecting a software artifact. An example is a change that is small in scale. The author may want to consider combining the preliminary design and detailed design inspections. Guidelines for determining when multiple inspections could be combined are:

- Combine the preliminary design and detailed design inspections when an update is made to an existing module and the preliminary design is not affected.
- Combine the detailed design and code inspections when the detailed design is done at a very detailed level and is expressed in the target language.

The moderator should be certified to run an inspection. Certification should consist of a minimum of 1 day of training in the inspection process and the specific role of the moderator within that process. In addition, newly trained moderators should be observed by an experienced moderator for the first two to three inspections. Other participants in an inspection should be given a 1-hour overview of the inspection process.

5.4.1.1.6 Parent Template(s)

<Field is Empty>

5.4.1.1.7 Child Templates...

5.4.1.1.8 Unique ID

INS_PLAN

5.4.1.1.8.1 Child Type

Composition

5.4.1.1.8.2 Child Tailoring

Required

5.4.1.1.8.3 Child Proximity

Local

5.4.1.1.9 Unique ID

INS_OVR

5.4.1.1.9.1 Child Type

Composition

5.4.1.1.9.2 Child Tailoring

Suggested

5.4.1.1.9.3 Child Proximity

Local

5.4.1.1.10 Unique ID

INS_PREP

5.4.1.1.10.1 Child Type

Composition

5.4.1.1.10.2 Child Tailoring

Recommended

5.4.1.1.10.3 Child Proximity

Local

5.4.1.1.11 Unique ID

INS_MTG

5.4.1.1.11.1 Child Type

Composition

5.4.1.1.11.2 Child Tailoring

Recommended

5.4.1.1.11.3 Child Proximity

Local

5.4.1.1.12 Unique ID

INS_REWK

5.4.1.1.12.1 Child Type

Composition

5.4.1.1.12.2 Child Tailoring

Suggested

5.4.1.1.12.3 Child Proximity

Local

5.4.1.1.13 Unique ID

INS_FWUP

5.4.1.1.13.1 Child Type

Composition

5.4.1.1.13.2 Child Tailoring

Suggested

5.4.1.1.13.3 Child Proximity

Local

5.4.1.1.14 Used Within...

5.4.1.1.15 Output Object Name

Peer_Review_Procedure_Manual

5.4.1.1.15.1 Output Object Type

Procedure_Manual

5.4.1.1.15.2 Include In Next Build

Yes

5.4.1.1.16 Transition Text...

5.4.1.1.17 Output Object Name

Peer_Review_Procedure_Manual

5.4.1.1.17.1 Leading Transition Text

The technical office has decided that the Peer Review process used internally will be a variant of the Fagan formal inspection method. Comments and recommendations for improving either this material in particular, or the peer review process in general, should be forwarded through your local SEPG representative.

5.4.1.1.17.2 Trailing Transition Text

<Field is Empty>

5.4.1.1.18 Activity Criteria and Process...

5.4.1.1.19 Entry Criteria

The author or project manager has selected a trained moderator for the preliminary design, detailed design, or code inspection.

The moderator has agreed to moderate the indicated inspection.

The moderator has been notified by the author that the preliminary design, detailed design, or code is completed, baselined, and is ready for examination.

5.4.1.1.20 Internal Process

There are seven parts to the inspection process. Six of these parts are required. These include: planning, preparation, inspection, causal analysis, rework, and follow-up. The optional part is the overview. The following sections indicate the entrance criteria, procedure, and exit criteria for each of these parts.

5.4.1.1.21 Exit Criteria

The Inspection Close Memo has been completed and distributed.

5.4.1.1.22 Invariants

The product being inspected continues to have purpose.

5.4.1.1.23 Related Products...

5.4.1.1.24 Unique ID

IIP

5.4.1.1.24.1 Usage

Before Starting

5.4.1.1.24.2 Impact

Left Unchanged

5.4.1.1.24.3 Tailoring Options

Required

5.4.1.1.24.4 Elaboration Text

<Field is Empty>

5.4.1.1.25 Unique ID

MEMO

5.4.1.1.25.1 Usage

While Underway

5.4.1.1.25.2 Impact

Created

5.4.1.1.25.3 Tailoring Options

Recommended

5.4.1.1.25.4 Elaboration Text

<Field is Empty>

5.4.1.1.26 Unique ID

RMET

5.4.1.1.26.1 Usage

Before Ending

5.4.1.1.26.2 Impact

Created

5.4.1.1.26.3 Tailoring Options

Required

5.4.1.1.26.4 Elaboration Text

<Field is Empty>

5.4.1.1.27 Related Supports...

5.4.1.1.28 Unique ID

PRT

5.4.1.1.28.1 Usage

While Underway

5.4.1.1.28.2 Tailoring Options

Required

5.4.1.1.28.3 Elaboration Text

<Field is Empty>

5.4.1.1.29 Unique ID

RCO

5.4.1.1.29.1 Usage

Before Starting

5.4.1.1.29.2 Tailoring Options

Required

5.4.1.1.29.3 Elaboration Text

<Field is Empty>

5.4.1.2 Activity Template: Inspection Planning

5.4.1.2.1 Name

Inspection Planning

5.4.1.2.2 Unique ID

INS_PLAN

5.4.1.2.3 Brief Description

The Inspection Planning activity plans the time and resources of the inspection process.

5.4.1.2.4 Overview Description

During the planning stage, the author (i.e., owner) gives a copy of the material to be inspected to the moderator, who verifies the inspection readiness of the artifact. The moderator typically completes this assessment by using an entry criteria checklist suitable for the artifact type being inspected. If the artifact is ready to be inspected, the moderator and the author of the artifact identify relevant reference material, a checklist of likely defects (i.e., a preparation checklist), the participants and their respective roles, and the inspection schedule. The moderator generates the preparation checklist, which is suitable for the artifact type being inspected, by modifying a generic checklist.

The moderator uses planning guidelines, suitable for the artifact type being inspected, to aid in determining the inspection schedule. These guidelines include limiting the length of a single inspection meeting to 2 hours and not scheduling more than 4 hours per day to inspection activities. The moderator can derive guidelines to determine the duration of the overview, preparation, and inspection meeting activities from the following:

5.4.1.2.5 Summary Description

<Field is Empty>

5.4.1.2.6 Parent Template(s)

INS

5.4.1.2.7 Child Templates...

<Fields are Empty>

5.4.1.2.8 Used Within...

5.4.1.2.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.1.2.8.1.1 Output Object Type

Procedure_Manual

5.4.1.2.8.1.2 Include In Next Build

Yes

5.4.1.2.9 Transition Text...

<Fields are Empty>

5.4.1.2.10 Activity Criteria and Process...

5.4.1.2.10.1 Entry Criteria

The author submits a completed entry criteria checklist with the inspection package to the moderator. Refer to the appropriate procedure for the details of the entrance criteria checklists.

5.4.1.2.10.2 Internal Process

The moderator examines the materials to determine:

- The correct level of detail.

- Whether it is an appropriate amount to inspect.
- The availability of appropriate reference documents.

The moderator calculates the time required for the overview meeting, preparation time, and inspection meeting by dividing the size of the document to be inspected, in pages, or the code to be inspected, in lines of code, by the following estimates (these estimates are based on data related to an inspection process used in another organization and will be revised as data is collected):

- Overview. An overview should be limited to a single two-hour meeting. If an overview longer than two hours is indicated by the length of the material, the moderator and the author should examine the material to divide it into separate inspection pieces. If the material indicates that an overview can be completed in a half-hour or less, the overview can be held at the beginning of the inspection meeting.
- Preparation. An estimate for preparation rates for preliminary design material is 12 to 15 pages per hour; for detailed design material, 8 to 12 pages per hour; and for code, 200 noncomment source statements per hour.
- Inspection. An estimate for inspection rates for preliminary design material is 10 to 12 pages per hour; for detailed design material, 8 to 10 pages per hour; and for code, 100 to 150 noncomment source statements per hour.

The moderator, project manager, or author contacts the inspection participants, tells them what is expected, and solicits available dates.

If an overview meeting is desired, it is scheduled by the moderator.

The moderator schedules the inspection using the estimated inspection meeting time computed earlier. An inspection may require multiple meetings. The length of a single inspection meeting should not exceed 4 hours in a given day. For each 4-hour meeting, it should be organized into a 2-hour meeting, a half-hour break, and then the remaining 2 hours.

The moderator or author (with approval of the moderator) distributes the inspection package consisting of the reference and inspection materials to the participants after appending the following:

- The Inspection Invitation Memo which identifies the type of inspection, time, location, time charge number, participants, and job assignments for the inspection
- Trivial Error Log

5.4.1.2.10.3 Exit Criteria

If an overview meeting is indicated, the overview meeting has been scheduled.

The inspection meeting has been scheduled.

The inspection package has been distributed to all inspection participants.

5.4.1.2.10.4 Invariants

<Field is Empty>

5.4.1.2.11 Related Products...

5.4.1.2.11.1 Unique ID

RMET

5.4.1.2.11.1.1 Usage

Before Starting

5.4.1.2.11.1.2 Impact

Left Unchanged

5.4.1.2.11.1.3 Tailoring Options

Required

5.4.1.2.11.1.4 Elaboration Text

<Field is Empty>

5.4.1.2.11.2 Unique ID

MEMO_INV

5.4.1.2.11.2.1 Usage

Before Ending

5.4.1.2.11.2.2 Impact

Created

5.4.1.2.11.2.3 Tailoring Options

Required

5.4.1.2.11.2.4 Elaboration Text

<Field is Empty>

5.4.1.2.12 Related Supports...

5.4.1.2.12.1 Unique ID

RCO

5.4.1.2.12.1.1 Usage

Before Starting

5.4.1.2.12.1.2 Tailoring Options

Required

5.4.1.2.12.1.3 Elaboration Text

<Field is Empty>

5.4.1.2.12.2 Unique ID

PRT_MOD

5.4.1.2.12.2.1 Usage

Before Starting

5.4.1.2.12.2.2 Tailoring Options

Required

5.4.1.2.12.2.3 Elaboration Text

<Field is Empty>

5.4.1.2.12.3 Unique ID

PRT_DEV

5.4.1.2.12.3.1 Usage

Before Starting

5.4.1.2.12.3.2 Tailoring Options

Suggested

5.4.1.2.12.3.3 Elaboration Text

<Field is Empty>

5.4.1.3 Activity Template: Inspection Overview

5.4.1.3.1 Name

Inspection Overview

5.4.1.3.2 Unique ID

INS_OVR

5.4.1.3.3 Brief Description

The Inspection Overview activity provides the context of the Inspection activity and explanation of the product to be inspected.

5.4.1.3.4 Overview Description

The overview consists of a presentation, by the author, of a general description of the inspection material to the remaining participants. Additionally, the moderator may use the overview meeting to emphasize the purpose of the inspection method and to ensure that the participants understand their roles and responsibilities. This meeting is typically held prior to the inspection meeting.

5.4.1.3.5 Summary Description

<Field is Empty>

5.4.1.3.6 Parent Template(s)

INS

5.4.1.3.7 Child Templates...

<Fields are Empty>

5.4.1.3.8 Used Within...

5.4.1.3.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.1.3.8.1.1 Output Object Type

Procedure_Manual

5.4.1.3.8.1.2 Include In Next Build

Yes

5.4.1.3.9 Transition Text...

<Fields are Empty>

5.4.1.3.10 Activity Criteria and Process...

5.4.1.3.10.1 Entry Criteria

The planning stage has been completed.

All key inspectors are present.

5.4.1.3.10.2 Internal Process

The moderator runs the meeting.

The author presents his design, addressing the following:

- The purpose of the functional changes/additions
- The external interfaces
- The relation between the modules, as a function, to the rest of the software component and to the software product

5.4.1.3.10.3 Exit Criteria

The overview has been presented.

5.4.1.3.10.4 Invariants

<Field is Empty>

5.4.1.3.11 Related Products...

5.4.1.3.11.1 Unique ID

IIP

5.4.1.3.11.1.1 Usage

Before Starting

5.4.1.3.11.1.2 Impact

Left Unchanged

5.4.1.3.11.1.3 Tailoring Options

Required

5.4.1.3.11.1.4 Elaboration Text

<Field is Empty>

5.4.1.3.12 Related Supports...

5.4.1.3.12.1 Unique ID

PRT

5.4.1.3.12.1.1 Usage

Before Starting

5.4.1.3.12.1.2 Tailoring Options

Required

5.4.1.3.12.1.3 Elaboration Text

<Field is Empty>

5.4.1.4 Activity Template: Inspection Preparation

5.4.1.4.1 Name

Inspection Preparation

5.4.1.4.2 Unique ID

INS_PREP

5.4.1.4.3 Brief Description

The Inspection Preparation activity consists of each inspector individually reviewing the product.

5.4.1.4.4 Overview Description

During the preparation stage, each inspector individually reviews the inspection material. During this individual review, he verifies the artifact for technical accuracy, fulfillment of requirements, and adherence to standards and conventions. The inspector records any defects discovered during this activity stage. He may also develop questions which may lead to the discovery of additional defects during the inspection meeting. The moderator ensures that adequate time has been allocated for proper preparation and assists the inspectors, as needed, in their preparation activity.

5.4.1.4.5 Summary Description

<Field is Empty>

5.4.1.4.6 Parent Template(s)

INS

5.4.1.4.7 Child Templates...

<Fields are Empty>

5.4.1.4.8 Used Within...

5.4.1.4.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.1.4.8.1.1 Output Object Type

Procedure_Manual

5.4.1.4.8.1.2 Include In Next Build

Yes

5.4.1.4.9 Transition Text...

<Fields are Empty>

5.4.1.4.10 Activity Criteria and Process...

5.4.1.4.10.1 Entry Criteria

The planning stage has been completed if a separate overview is not planned. Otherwise, the overview has been completed.

5.4.1.4.10.2 Internal Process

The moderator ensures that each inspector has received the inspection package. This may be done by phone, electronic mail, or personal visit.

The moderator ensures that sufficient preparation time is available for each inspector.

Each participant prepares for the inspection. During preparation, trivial errors are documented on the Trivial Error Log. Refer to the appropriate procedure for guidelines for reviewing the inspection material.

5.4.1.4.10.3 Exit Criteria

Each inspector has the materials. knows his role, has had sufficient time to prepare, knows the place and time of the inspection, and has a copy of the Trivial Error Log.

The moderator determines that all key inspectors are prepared for the inspection.

5.4.1.4.10.4 Invariants

<Field is Empty>

5.4.1.4.11 Related Products...

5.4.1.4.11.1 Unique ID

IIP

5.4.1.4.11.1.1 Usage

Before Starting

5.4.1.4.11.1.2 Impact

Left Unchanged

5.4.1.4.11.1.3 Tailoring Options

Required

5.4.1.4.11.1.4 Elaboration Text

<Field is Empty>

5.4.1.4.12 Related Supports...

5.4.1.4.12.1 Unique ID

PRT

5.4.1.4.12.1.1 Usage

Before Starting

5.4.1.4.12.1.2 Tailoring Options

Required

5.4.1.4.12.1.3 Elaboration Text

<Field is Empty>

5.4.2 Peer Review Product Templates

Task 5 MPDM involves capturing additional information about your products by using the templates.

Table 5-2 shows the example product templates in this section. (Again, the full set of templates is shown in Appendix A, Volume 2.)

Table 5-2. Example Product Templates

Product Name	Unique Identifier
Review Memos	MEMO
Inspection Invitation Memorandum	MEMO_INV
Review Metrics	RMET

5.4.2.1 Product Template: Review Memos

5.4.2.1.1 Name

Review Memos

5.4.2.1.2 Unique ID

MEMO

5.4.2.1.3 Brief Description

This product is the set of memorandums that are used to support the inspection process.

5.4.2.1.4 Overview Description

The inspection process is facilitated by the use of memorandums. These serve three major purposes: (1) to convey information about upcoming inspection activities, (2) to serve as a type of checklist for those participating in the inspection process, (3) as an informal means to temporarily report metric information.

There are two primary types of memorandums currently in use: (1) The Invitation memorandum, and (2) the Results memorandum.

5.4.2.1.5 Summary Description

In addition to these two types of memorandums, moderators and review coordinators are encouraged to develop and use other memorandums if it appears that they will improve the efficiency or effectiveness of the inspection process.

If additional memorandums are developed, they should be taken to the causal analysis meeting for consideration and possible recommendation as new inspection support products.

5.4.2.1.6 Parent Template(s)

<Field is Empty>

5.4.2.1.7 Child Templates...

5.4.2.1.7.1 Unique ID

MEMO_INV

5.4.2.1.7.1.1 Child Type

Specialization

5.4.2.1.7.1.2 Child Tailoring

Recommended

5.4.2.1.7.1.3 Child Proximity

Local

5.4.2.1.7.2 Unique ID

MEMO_REV

5.4.2.1.7.2.1 Child Type

Specialization

5.4.2.1.7.2.2 Child Tailoring

Recommended

5.4.2.1.7.2.3 Child Proximity

Local

5.4.2.1.8 Used Within...

5.4.2.1.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.2.1.8.1.1 Output Object Type

Procedure_Manual

5.4.2.1.8.1.2 Include In Next Build

Yes

5.4.2.1.9 Transition Text...

5.4.2.1.9.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.2.1.9.1.1 Leading Transition Text

<Field is Empty>

5.4.2.1.9.1.2 Trailing Transition Text

Following are specific details about these two types of memorandums.

5.4.2.1.10 Related Activities

INS

5.4.2.2 Product Template: Inspection Invitation Memorandum

5.4.2.2.1 Name

Inspection Invitation Memorandum

5.4.2.2.2 Unique ID

MEMO_INV

5.4.2.2.3 Brief Description

This memorandum is used to inform participants of an upcoming inspection.

5.4.2.2.4 Overview Description

The Invitation memorandum informs all participants that they have been selected to participate in a coming inspection. The memorandum includes the name of the artifact to be inspected, the amount of time the reviewer is expected to spend preparing for the inspection, the date that the inspection material will be distributed, the date and time of the inspection, and the role each person in the review team is expected to play.

5.4.2.2.5 Summary Description

<Field is Empty>

5.4.2.2.6 Parent Template(s)

MEMO

5.4.2.2.7 Child Templates...

<Fields are Empty>

5.4.2.2.8 Used Within...

5.4.2.2.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.2.2.8.1.1 Output Object Type

Procedure_Manual

5.4.2.2.8.1.2 Include In Next Build

Yes

5.4.2.2.9 Transition Text...

5.4.2.2.9.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.2.2.9.1.1 Leading Transition Text

<Field is Empty>

5.4.2.2.9.1.2 Trailing Transition Text

For an example of what this memorandum looks like, refer to the appendixes.

5.4.2.2.10 Related Activities

INS_PLAN

5.4.2.3 Product Template: Review Metrics

5.4.2.3.1 Name

Review Metrics

5.4.2.3.2 Unique ID

RMET

5.4.2.3.3 Brief Description

This is a hard-copy report summarizing the inspection results.

5.4.2.3.4 Overview Description

The review metrics product is a report that contains all the metric information gathered during the inspection process. The primary contents of this report includes:

- Number of defects (by type and severity) found during preparation
- Number of defects (by type and severity) found during the inspection meeting
- Number of people that prepared for the inspection
- Total time spent in preparation
- Number of people that attended the inspection meeting
- Inspection meeting start time
- Inspection meeting completion time
- Total staff-time spent reviewing the product within the inspection meeting
- Estimated rework time
- Actual rework time
- Names and roles of all participants

5.4.2.3.5 Summary Description

<Field is Empty>

5.4.2.3.6 Parent Template(s)

<Field is Empty>

5.4.2.3.7 Child Templates...

<Fields are Empty>

5.4.2.3.8 Used Within...

5.4.2.3.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.2.3.8.1.1 Output Object Type

Procedure_Manual

5.4.2.3.8.1.2 Include In Next Build

Yes

5.4.2.3.9 Transition Text...

5.4.2.3.9.1 Output Object Name

Peer_Review_Training_Material

5.4.2.3.9.1.1 Leading Transition Text

<Field is Empty>

5.4.2.3.9.1.2 Trailing Transition Text

On a separate piece of paper, describe at least five additional metrics that might be important to collect during the inspection process. Explain both how and when they would be collected and for what they would be used. Please spend about 20 minutes on this exercise.

5.4.2.3.10 Related Activities

INA

5.4.3 Peer Review Role Templates

Task 7 of MPDM involves capturing information about the primary roles which participate in the process.

Table 5-3 shows example role templates in this section.

Table 5-3. Example Role Templates

Role Name	Unique Identifier
Peer Review Team	PRT
Inspectors	PRT_INS
Key Inspector	PRT_INS_KEY
Review Coordinator	RCO

5.4.3.1 Role Template: Peer Review Team

5.4.3.1.1 Name

Peer Review Team

5.4.3.1.2 Unique ID

PRT

5.4.3.1.3 Brief Description

The Peer Review Team participates in and is responsible for all phases of the Peer Review process.

5.4.3.1.4 Overview Description

Peer Review Teams conduct reviews of products (sometimes several times) before those products are placed into Configuration Management. The team typically contains three to six people who have one or more of the roles of: Moderator, Scribe, Reader, Key and Regular Inspectors, and Developer. Note that the developer of the product being inspected serves as part of the review team.

5.4.3.1.5 Summary Description

<Field is Empty>

5.4.3.1.6 Parent Template(s)

<Field is Empty>

5.4.3.1.7 Child Templates...

5.4.3.1.7.1 Unique ID

PRT_MOD

5.4.3.1.7.1.1 Child Type

Composition

5.4.3.1.7.1.2 Child Tailoring

Required

5.4.3.1.7.1.3 Child Proximity

Local

5.4.3.1.7.2 Unique ID

PRT_RDR

5.4.3.1.7.2.1 Child Type

Composition

5.4.3.1.7.2.2 Child Tailoring

Required

5.4.3.1.7.2.3 Child Proximity

Local

5.4.3.1.7.3 Unique ID

PRT_SCB

5.4.3.1.7.3.1 Child Type

Composition

5.4.3.1.7.3.2 Child Tailoring

Required

5.4.3.1.7.3.3 Child Proximity

Local

5.4.3.1.7.4 Unique ID

PRT_INS

5.4.3.1.7.4.1 Child Type

Composition

5.4.3.1.7.4.2 Child Tailoring

Required

5.4.3.1.7.4.3 Child Proximity

Local

5.4.3.1.7.5 Unique ID

PRT_DEV

5.4.3.1.7.5.1 Child Type

Composition

5.4.3.1.7.5.2 Child Tailoring

Suggested

5.4.3.1.7.5.3 Child Proximity

Local

5.4.3.1.8 Used Within...

5.4.3.1.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.3.1.8.1.1 Output Object Type

Procedure_Manual

5.4.3.1.8.1.2 Include In Next Build

Yes

5.4.3.1.9 Transition Text...

5.4.3.1.9.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.3.1.9.1.1 Leading Transition Text

<Field is Empty>

5.4.3.1.9.1.2 Trailing Transition Text

Details about each of these roles are presented below.

5.4.3.1.10 Supported Activities

INA
CA

5.4.3.1.11 Reports To...

<Not Applicable>

5.4.3.1.12 Reported To By...

<Not Applicable>

5.4.3.2 Role Template: Inspectors

5.4.3.2.1 Name

Inspectors

5.4.3.2.2 Unique ID

PRT_INS

5.4.3.2.3 Brief Description

Inspectors evaluate material against both formal and informal standards to identify and log defects.

5.4.3.2.4 Overview Description

Each inspector is responsible for reviewing the material during the preparation stage and for participating in the inspection meeting. Their goal during the preparation activity and the inspection meeting is to find defects in the artifact or material being inspected. Inspectors are considered to be of two types: key inspectors and regular inspectors.

5.4.3.2.5 Summary Description

In addition to key and regular inspectors there is also the informal role of "guest inspector." Although rarely part of the process, a guest inspector is someone who does not participate in any of the meetings. Instead, guest inspectors evaluate the product and send the results of their evaluation to the moderator.

5.4.3.2.6 Parent Template(s)

PRT

5.4.3.2.7 Child Templates...

5.4.3.2.7.1 Unique ID

PRT_INS_KEY

5.4.3.2.7.1.1 Child Type

Specialization

5.4.3.2.7.1.2 Child Tailoring

Required

5.4.3.2.7.1.3 Child Proximity

Local

5.4.3.2.7.2 Unique ID

PRT_INS_REG

5.4.3.2.7.2.1 Child Type

Specialization

5.4.3.2.7.2.2 Child Tailoring

Suggested

5.4.3.2.7.2.3 Child Proximity

Local

5.4.3.2.8 Used Within...

5.4.3.2.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.3.2.8.1.1 Output Object Type

Procedure_Manual

5.4.3.2.8.1.2 Include In Next Build

Yes

5.4.3.2.9 Transition Text...

5.4.3.2.9.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.3.2.9.1.1 Leading Transition Text

<Field is Empty>

5.4.3.2.9.1.2 Trailing Transition Text

<Field is Empty>

5.4.3.2.9.2 Output Object Name

Peer_Review_Training_Material

5.4.3.2.9.2.1 Leading Transition Text

<Field is Empty>

5.4.3.2.9.2.2 Trailing Transition Text

On a separate piece of paper, briefly describe one or two additional types of inspectors the might be useful to have in the Peer Review process. Please spend about 10 minutes on this exercise.

5.4.3.2.10 Supported Activities

INS_OVR
INS_PREP
INS_MTG

5.4.3.2.11 Reports To...

5.4.3.2.11.1 Unique ID

PRT_MOD

5.4.3.2.11.1.1 Elaboration Text

<Field is Empty>

5.4.3.2.12 Reported To By...

<Field is Empty>

5.4.3.3 Role Template: Key Inspector

5.4.3.3.1 Name

Key Inspector

5.4.3.3.2 Unique ID

PRT_INS_KEY

5.4.3.3.3 Brief Description

Key inspectors are inspectors that are required at the inspection meeting.

5.4.3.3.4 Overview Description

Key inspectors have the primary responsibility for inspecting the product or artifact under review. Generally, candidates are identified to be key inspectors due to particularly high levels of experience, insight, or ability that they can contribute to reviewing the artifact. Generally, key inspectors should have different specialty areas so as to provide greater scope in evaluating the artifact.

5.4.3.3.5 Summary Description

<Field is Empty>

5.4.3.3.6 Parent Template(s)

PRT_INS

5.4.3.3.7 Child Templates...

<Fields are Empty>

5.4.3.3.8 Used Within...

5.4.3.3.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.3.3.8.1.1 Output Object Type

Procedure_Manual

5.4.3.3.8.1.2 Include In Next Build

Yes

5.4.3.3.9 Transition Text...

<Fields are Empty>

5.4.3.3.10 Supported Activities

<Field is Empty>

5.4.3.3.11 Reports To...

<Fields are Empty>

5.4.3.3.12 Reported To By

<Field is Empty>

5.4.3.4 Role Template: Review Coordinator

5.4.3.4.1 Name

Review Coordinator

5.4.3.4.2 Unique ID

RCO

5.4.3.4.3 Brief Description

Review coordinators are responsible for organizing and overseeing reviews.

5.4.3.4.4 Overview Description

The primary responsibility of review coordinators is to see that all planned reviews are conducted according to schedule, that the necessary output products are produced and archived, and that reviews are conducted in an efficient and effective manner. One key area of responsibility is ensuring that review work is evenly and fairly divided among all those who are authorized to participate. As a secondary responsibility, review coordinators also ensure that all participants have the necessary training, or if not, that arrangements are made so that training occurs.

5.4.3.4.5 Summary Description

<Field is Empty>

5.4.3.4.6 Parent Template(s)

<Field is Empty>

5.4.3.4.7 Child Templates...

5.4.3.4.8 Used Within...

5.4.3.4.8.1 Output Object Name

Peer_Review_Procedure_Manual

5.4.3.4.8.1.1 Output Object Type

Procedure_Manual

5.4.3.4.8.1.2 Include In Next Build

Yes

5.4.3.4.9 Transition Text...

<Fields are Empty>

5.4.3.4.10 Supported Activities

INS
CA

5.4.3.4.11 Reports To...

<Fields are Empty>

5.4.3.4.12 Reported To By

PRT_MOD

5.5 EXTRACT ALGORITHMS

Task 8 in MPDM involves generating process end-products from the process model. As discussed at the end of Section 4, this involves the following steps:

1. Select a primary "view" or orientation for your end-product.
2. Define how you will navigate the database to find and select templates in accordance with your primary view.
3. Select the auxiliary information you want, and when you want to export it.

The algorithm in Section 5.5.1 is presented using structured English. This example shows how to navigate through and extract from the database, exporting fields for building an activity-oriented guidebook. You should start with a simple navigation and extract algorithm and then evolve the algorithm over time.

Not all of the subroutines are elaborated. Depending on the environment you select, you may not need to, for instance, write an algorithm for "Sort_List_of_Related_Roles()". Many environments provide extensive support for sorting, building lists, etc. Section 5.5.1 is intended to communicate the overall approach to navigation and extracting algorithms.

5.5.1 Algorithm

```
Control_Algorithm is:
{
    Find the Root activity template
    Print_Activity_and_Related_Information()
    Export Fixed Text: "SUBACTIVITIES"
    Seek_Show_All_Children()
    Print_Activity_Summary_Information()
    Export Fixed Text: "APPENDIX A: PRODUCTS"
    Build_List_of_Related_Products()
    Sort_List_of_Related_Products()
    Show_Listed_Product_Details()
    Export Fixed Text: "APPENDIX B: ROLES"
    Build_List_of_Related_Roles()
    Sort_List_of_Related_Roles()
    Show_Listed_Role_Details()
}

Seek_Show_All_Children is:
{
    While another child exists
    {
        Select next child
        Print_Activity_and_Related_Information()
        Export Fixed Text: "SUBACTIVITIES"
        Seek_Show_All_Children() /* recursive call */
        Print_Activity_Summary_Information()
    }
}

Print_Activity_and_Related_Information is:
{
    Export_Transition(LEADING, GRAPHIC)
    Export_Transition(LEADING, TEXT)
    Export Activity Name
    Export Activity Overview Description
    Export Activity Entry Criteria
    Export Activity Internal Process
    Export Activity Exit Criteria
    Export Activity Invariants
    While another Related_Product Exists
    {
        Get next PRODUCT_ID
        Show_Product_Details(PRODUCT_ID)
        Export Elaboration Text
    }
    While another Supporting_Role Exists
    {
        Get next ROLE_ID
        Show_Role_Details(ROLE_ID)
        Export Elaboration Text
    }
}

Print_Activity_Summary_Information is:
{
    Export Fixed Text: "SUMMARY"
    Export Summary Description
    Export Fixed Text: "REQUIREMENTS TRACEABILITY"
    While another Requirements Document Title exists
    {
        Get next Requirements Title
        Export Requirements Title
        While another Requirement Unique ID exists under this title
        {
            Get next Requirement Unique ID
            Export Requirement Unique ID
            Export Requirement Level of Satisfaction
        }
    }
    Export Fixed Text: "ADDITIONAL INFORMATION"
    While another Title available for additional information
    {
        Get next Title
        Export Title Name
        While another Section available under Title
        {
            Get next Section Name
            Export Section Name
            While another Pages reference available under Section
            {
                Get next Pages range
                Export Pages range
            }
        }
    }
    Export_Transition(TRAILING, TEXT)
    Export_Transition(TRAILING, GRAPHIC)
}

Show_Product_Details(PRODUCT_ID)
{
    Find Data For PRODUCT_ID
    Export Product Name
    Export Brief Description
    Export Usage
    Export Impact
    Export Tailoring
}
```

5.5.2 Style of Output

The following example is intended to convey a general idea about the order, and hence appearance, that would result from running an algorithm similar to the one presented in Section 5.5.1. Keep in mind that you can significantly change the appearance, scope, and level of detail in a guidebook (or any other process end-product) by changing the way your extract algorithm navigates the database and by changing which fields are selected for export.

The preceding algorithm would yield a document whose information was ordered along the following lines. The example is somewhat lengthy, but shows navigation to several levels, inclusion of product and role information, and example ordering of information for appendixes.

```
Leading Transition Text and Graphics from Root Activity
LEVEL 0 ACTIVITY
Root Activity Name
OVERVIEW
Root Activity Overview Description
CRITERIA AND PROCESS
Root Activity Entry/Internal/Exit/Invariant Information
RELATED PRODUCTS
Related Product 1 Information (Name, Brief Description, and Elaboration Text)
Related Product 2 Information
Related Product N Information
RELATED ROLES
Supporting Role 1 Information (Name, Brief Description, and Elaboration Text)
Supporting Role 2 Information
Supporting Role N Information
SUBACTIVITIES
    Child 1 Leading Transition Text and Graphics
    LEVEL 1 ACTIVITY
    Child 1 Name
    Child 1 OVERVIEW
    Child 1 Overview Description
    Child 1 CRITERIA AND PROCESS
    Child 1 Entry/Internal/Exit/Invariant Information
    Child 1 RELATED PRODUCTS
    Child 1 Related Product 1 Information
    Child 1 Related Product N Information
    Child 1 RELATED ROLES
    Child 1 Supporting Role 1 Information
    Child 1 Supporting Role N Information
    Child 1 SUBACTIVITIES
        Child 1's, Child A Leading Transition Text and Graphics
        LEVEL 2 ACTIVITY
        Child 1's, Child A Name
```

Child 1's, Child A OVERVIEW
Child 1's, Child A Overview Description
Child 1's, Child A CRITERIA AND PROCESS
Child 1's, Child A Entry/Internal/Exit/Invariant Information
Child 1's, Child A RELATED PRODUCTS
Child 1's, Child A Related Product Information
Child 1's, Child A RELATED ROLES
Child 1's, Child A Supporting Role Information
Child 1's, Child A SUBACTIVITIES
Child 1's, Child A SUMMARY
Child 1's, Child A Summary Description
Child 1's, Child A REQUIREMENTS TRACEABILITY
Child 1's, Child A Requirements Document Titles/Requirements/Levels of Satisfaction
Child 1's, Child A ADDITIONAL INFORMATION
Child 1's, Child A Additional Information Source Titles/Sections/Pages
Child 1's, Child A Trailing Transition Text and Graphics
Child 1's, Child A Leading Transition Text and Graphics
LEVEL 2 ACTIVITY
Child 1's, Child B Name
Child 1's, Child B OVERVIEW
Child 1's, Child B Overview Description
Child 1's, Child B CRITERIA AND PROCESS
Child 1's, Child B Entry/Internal/Exit/Invariant Information
Child 1's, Child B RELATED PRODUCTS
Child 1's, Child B Related Product Information
Child 1's, Child B RELATED ROLES
Child 1's, Child B Supporting Role Information
Child 1's, Child B SUBACTIVITIES
Child 1's, Child B SUMMARY
Child 1's, Child B Summary Description
Child 1's, Child B REQUIREMENTS TRACEABILITY
Child 1's, Child B Requirements Document Titles/Requirements/Levels of Satisfaction
Child 1's, Child B ADDITIONAL INFORMATION
Child 1's, Child B Additional Information Source Titles/Sections/Pages
Child 1's, Child B Trailing Transition Text and Graphics

Child 1 SUMMARY
Child 1 Summary Description
Child 1 REQUIREMENTS TRACEABILITY
Child 1 Requirements Document Titles/Requirements/Levels of Satisfaction
Child 1 ADDITIONAL INFORMATION
Child 1 Additional Information Source Titles/Sections/Pages
Child 1 Trailing Transition Text and Graphics
Child 2 Leading Transition Text and Graphics
LEVEL 1 ACTIVITY
Child 2 Name
Child 2 OVERVIEW
Child 2 Overview Description

Child 2 CRITERIA AND PROCESS
Child 2 Entry/Internal/Exit/Invariant Information
Child 2 RELATED PRODUCTS
Child 2 Related Product 1 Information
Child 2 Related Product N Information
Child 2 RELATED ROLES
Child 2 Supporting Role 1 Information
Child 2 Supporting Role N Information
Child 2 SUBACTIVITIES

> Child 2's, Child A Leading Transition Text and Graphics
> LEVEL 2 ACTIVITY
> Child 2's, Child A Name
> Child 2's, Child A OVERVIEW
> Child 2's, Child A Overview Description
> Child 2's, Child A CRITERIA AND PROCESS
> Child 2's, Child A Entry/Internal/Exit/Invariant Information
> Child 2's, Child A RELATED PRODUCTS
> Child 2's, Child A Related Product Information
> Child 2's, Child A RELATED ROLES
> Child 2's, Child A Supporting Role Information
> Child 2's, Child A SUBACTIVITIES
>
> > Child 2's, Child A, Subchild 1 Leading Transition Text and Graphics
> > LEVEL 3 ACTIVITY
> > Child 2's, Child A, Subchild 1 Name
> > Child 2's, Child A, Subchild 1 OVERVIEW
> > Child 2's, Child A, Subchild 1 Overview Description
> > Child 2's, Child A, Subchild 1 CRITERIA AND PROCESS
> > Child 2's, Child A, Subchild 1 Entry/Internal/Exit/Invariant Information
> > Child 2's, Child A, Subchild 1 RELATED PRODUCTS
> > Child 2's, Child A, Subchild 1 Related Product Information
> > Child 2's, Child A, Subchild 1 RELATED ROLES
> > Child 2's, Child A, Subchild 1 Supporting Role Information
> > Child 2's, Child A, Subchild 1 SUBACTIVITIES
> > Child 2's, Child A, Subchild 1 SUMMARY
> > Child 2's, Child A, Subchild 1 Summary Description
> > Child 2's, Child A, Subchild 1 REQUIREMENTS TRACEABILITY
> > Child 2's, Child A, Subchild 1 Requirements Document Information
> > Child 2's, Child A, Subchild 1 ADDITIONAL INFORMATION
> > Child 2's, Child A, Subchild 1 Additional Information Source Titles/Sections/Pages
> > Child 2's, Child A, Subchild 1 Trailing Transition Text and Graphics
> > Child 2's, Child A, Subchild 2 Leading Transition Text and Graphics
> > LEVEL 3 ACTIVITY
> > Child 2's, Child A, Subchild 2 Name
> > Child 2's, Child A, Subchild 2 OVERVIEW
> > Child 2's, Child A, Subchild 2 Overview Description
> > Child 2's, Child A, Subchild 2 CRITERIA AND PROCESS
> > Child 2's, Child A, Subchild 2 Entry/Internal/Exit/Invariant Information

Child 2's, Child A, Subchild 2 RELATED PRODUCTS
Child 2's, Child A, Subchild 2 Related Product Information
Child 2's, Child A, Subchild 2 RELATED ROLES
Child 2's, Child A, Subchild 2 Supporting Role Information
Child 2's, Child A, Subchild 2 SUBACTIVITIES
Child 2's, Child A, Subchild 2 SUMMARY
Child 2's, Child A, Subchild 2 Summary Description
Child 2's, Child A, Subchild 2 REQUIREMENTS TRACEABILITY
Child 2's, Child A, Subchild 2 Requirements Document Information
Child 2's, Child A, Subchild 2 ADDITIONAL INFORMATION
Child 2's, Child A, Subchild 2 Additional Information Source Titles/Sections/Pages
Child 2's, Child A, Subchild 2 Trailing Transition Text and Graphics
Child 2's, Child A SUMMARY
Child 2's, Child A Summary Description
Child 2's, Child A REQUIREMENTS TRACEABILITY
Child 2's, Child A Requirements Document Titles/Requirements/Levels of Satisfaction
Child 2's, Child A ADDITIONAL INFORMATION
Child 2's, Child A Additional Information Source Titles/Sections/Pages
Child 2's, Child A Trailing Transition Text and Graphics
Child 2's, Child B Leading Transition Text and Graphics
LEVEL 2 ACTIVITY
Child 2's, Child B Name
Child 2's, Child B OVERVIEW
Child 2's, Child B Overview Description
Child 2's, Child B CRITERIA AND PROCESS
Child 2's, Child B Entry/Internal/Exit/Invariant Information
Child 2's, Child B RELATED PRODUCTS
Child 2's, Child B Related Product Information
Child 2's, Child B RELATED ROLES
Child 2's, Child B Supporting Role Information
Child 2's, Child B SUBACTIVITIES
Child 2's, Child B SUMMARY
Child 2's, Child B Summary Description
Child 2's, Child B REQUIREMENTS TRACEABILITY
Child 2's, Child B Requirements Document Titles/Requirements/Levels of Satisfaction
Child 2's, Child B ADDITIONAL INFORMATION
Child 2's, Child B Additional Information Source Titles/Sections/Pages
Child 2's, Child B Trailing Transition Text and Graphics
Child 2 SUMMARY
Child 2 Summary Description
Child 2 REQUIREMENTS TRACEABILITY
Child 2 Requirements Document Titles/Requirements/Levels of Satisfaction
Child 2 ADDITIONAL INFORMATION
Child 2 Additional Information Source Titles/Sections/Pages
Child 2 Trailing Transition Text and Graphics
SUMMARY
Root Activity Summary Description

REQUIREMENTS TRACEABILITY
Requirements Document Titles/Requirements/Levels of Satisfaction
ADDITIONAL INFORMATION
Additional Information Source Titles/Sections/Pages
Trailing Transition Text and Graphics from Root Activity

APPENDIX A: RELATED PRODUCTS
- Product 1 Leading Transition Text and Graphic
- PRODUCT DESCRIPTION
- Product 1 Name
- Product 1 Overview Description
- Product 1 Summary Description
- ACTIVITY REFERENCES
  - ACTIVITY DESCRIPTION
  - Activity A Name
  - Activity A Brief Description
  - Activity A Elaboration Text
  - ACTIVITY DESCRIPTION
  - Activity B Name
  - Activity B Brief Description
  - Activity B Elaboration Text
  - ACTIVITY DESCRIPTION
  - Activity X Name
  - Activity X Brief Description
  - Activity X Elaboration Text
- Product 1 Trailing Transition Text and Graphic
- Product 2 Leading Transition Text and Graphic
- PRODUCT DESCRIPTION
- Product 2 Name
- Product 2 Overview Description
- Product 2 Summary Description
- ACTIVITY REFERENCES
  - ACTIVITY DESCRIPTION
  - Activity D Name
  - Activity D Brief Description
  - Activity D Elaboration Text
  - ACTIVITY DESCRIPTION
  - Activity E Name
  - Activity E Brief Description
  - Activity E Elaboration Text
  - ACTIVITY DESCRIPTION
  - Activity X Name
  - Activity X Brief Description
  - Activity X Elaboration Text
- Product 2 Trailing Transition Text and Graphic
- Product N Leading Transition Text and Graphic
- PRODUCT DESCRIPTION

Product N Name
Product N Overview Description
Product N Summary Description
ACTIVITY REFERENCES
    ACTIVITY DESCRIPTION
    Activity A Name
    Activity A Brief Description
    Activity A Elaboration Text
    ACTIVITY DESCRIPTION
    Activity E Name
    Activity E Brief Description
    Activity E Elaboration Text
    ACTIVITY DESCRIPTION
    Activity F Name
    Activity F Brief Description
    Activity F Elaboration Text
    ACTIVITY DESCRIPTION
    Activity X Name
    Activity X Brief Description
    Activity X Elaboration Text
Product N Trailing Transition Text and Graphic

APPENDIX B: RELATED ROLES
Product 1 Leading Transition Text and Graphic
ROLE DESCRIPTION
Role 1 Name
Role 1 Overview Description
Role 1 Summary Description
ACTIVITY REFERENCES
    ACTIVITY DESCRIPTION
    Activity A Name
    Activity A Brief Description
    Activity A Elaboration Text
    ACTIVITY DESCRIPTION
    Activity B Name
    Activity B Brief Description
    Activity B Elaboration Text
    ACTIVITY DESCRIPTION
    Activity X Name
    Activity X Brief Description
    Activity X Elaboration Text
Role 1 Trailing Transition Text and Graphic
Role 2 Leading Transition Text and Graphic
ROLE DESCRIPTION
Role 2 Name
Role 2 Overview Description
Role 2 Summary Description

```
ACTIVITY REFERENCES
    ACTIVITY DESCRIPTION
    Activity D Name
    Activity D Brief Description
    Activity D Elaboration Text
    ACTIVITY DESCRIPTION
    Activity E Name
    Activity E Brief Description
    Activity E Elaboration Text
    ACTIVITY DESCRIPTION
    Activity X Name
    Activity X Brief Description
    Activity X Elaboration Text
Role 2 Trailing Transition Text and Graphic
Role N Leading Transition Text and Graphic
ROLE DESCRIPTION
Role N Name
Role N Overview Description
Role N Summary Description
ACTIVITY REFERENCES
    ACTIVITY DESCRIPTION
    Activity A Name
    Activity A Brief Description
    Activity A Elaboration Text
    ACTIVITY DESCRIPTION
    Activity E Name
    Activity E Brief Description
    Activity E Elaboration Text
    ACTIVITY DESCRIPTION
    Activity F Name
    Activity F Brief Description
    Activity F Elaboration Text
    ACTIVITY DESCRIPTION
    Activity X Name
    Activity X Brief Description
    Activity X Elaboration Text
Role N Trailing Transition Text and Graphic
```

5.6 OUTPUT EXAMPLE

Following is a brief example of the type of output that can be produced using an algorithm similar to that presented in Section 5.5.1. For space considerations, only a few pages are shown. Section 6 contains additional examples of formatted output (from a pilot project).

<< Start of Example >>

--- PEER REVIEW PROCESS ---

The technical office has decided that the Peer Review process used internally will be a variant of the Fagan formal inspection method. Comments and recommendations for improving either this material in particular, or the Peer Review process in general, should be forwarded through your local SEPG representative.

OVERVIEW OF PEER REVIEW PROCESS

The inspection method is one technique for the static verification of an artifact. Developed by Michael E. Fagan of IBM (Fagan 1984), the primary objective of this method is to detect defects in an artifact in an efficient and effective manner. Frequently referred to as a "formal inspection," you can use this method to verify **any** artifact generated during the development process, although considerable focus has been given to applying this method to preliminary design, detailed design, and code artifacts.

The inspection method consists of well-defined roles and activity stages. Associated with the roles are specific responsibilities. Additionally, entrance and exit criteria and objectives for each activity stage and mechanisms for recording and reporting defects ensure consistent application of this method.

The efficiency of this approach depends on focusing the participants on the overall objective of the method, which is defect detection. The effectiveness of this method depends on proper training of the participants and commitment by management to adhere to the prescribed process.

As stated earlier, you can use the inspection method to verify any artifact generated during the development process. This section, however, states the specific purpose for preliminary design, detailed design, and code inspections.

During the preliminary design phase, you hold preliminary design inspections to inspect the design documentation to verify that the requirements are satisfied and that the product will be maintainable, adaptable, and of high quality.

During the detailed design phase, you hold detailed design inspections to inspect the design documentation to verify that the design has been refined correctly, is defined to a level which allows coding to begin, and satisfies assigned product requirements. You inspect the design to ensure that it is maintainable, adaptable, and of high quality.

During the implementation phase, you hold code inspections to inspect the code to verify that it implements the detailed design correctly and completely.

The project team must baseline all reference material (i.e., specifications, requirements, documents) prior to the inspection meeting. For example, a preliminary design inspection cannot be held until they have baselined the requirements. An example of a requirements specification document is a system requirements specification (SRS).

PEER REVIEW CRITERIA AND PROCESS

Entry Criteria

The author or project manager has selected a trained moderator for the preliminary design, detailed design, or code inspection.

The moderator has agreed to moderate the indicated inspection.

The moderator has been notified by the author that the preliminary design, detailed design, or code is completed, baselined, and is ready for examination.

Internal Process

There are seven parts to the inspection process. Fix of these parts are required. These include: planning, preparation, inspection, causal analysis, rework, and follow-up. The optional part is the overview.

Exit Criteria

The Inspection Close Memo has been completed and distributed.

Invariants

The product being inspected continues to have purpose.

PEER REVIEW PROCESS RELATED PRODUCTS

Inspection Input Product

This is the product, item, or artifact to be inspected.
The Inspection Input Product is needed before starting this activity.
This activity leaves the Inspection Input Product unchanged.
This is a required product.

Review Memos

This product is the set of memorandums that are used to support the inspection process.
This product is needed sometime after the activity has started.
This product is created by this activity.
This is a recommended product; but you have the option to apply for a waiver.

Review Metrics

This is a hardcopy report summarizing the inspection results.
This product is produced before the activity can be considered complete.
This product is created by this activity.
This is a required product.

PEER REVIEW PROCESS RELATED ROLES

Peer Review Team

The Peer Review Team participates in and is responsible for all phases of the Peer Review Process.
This role is needed sometime after the activity has started.
This is a required role.

Review Coordinator

Review coordinators are responsible for organizing and overseeing reviews.
The Review Coordinator Role is needed before starting this activity.
This is a required role.

PEER REVIEW PROCESS SUBACTIVITIES

1.0 Activity Name: Inspection Planning

OVERVIEW OF INSPECTION PLANNING

During the planning stage, the author (i.e., owner) gives a copy of the material to be inspected to the moderator, who verifies the inspection readiness of the artifact. The moderator typically completes this assessment by using an entry criteria checklist suitable for the artifact type being inspected. If the artifact is ready to be inspected, the moderator and the author of the artifact identify relevant reference material, a checklist of likely defects (i.e., a preparation checklist), the participants and their respective roles, and the inspection schedule. The moderator generates the preparation checklist, which is suitable for the artifact type being inspected, by modifying a generic checklist.

The moderator uses planning guidelines, suitable for the artifact type being inspected, to aid in determining the inspection schedule. These guidelines include limiting the length of a single inspection meeting to two hours and not scheduling more than four hours per day to inspection activities. The moderator can derive guidelines to determine the duration of the overview, preparation, and inspection meeting activities from the following:

INSPECTION PLANNING CRITERIA AND PROCESS

Entry Criteria

The author submits a completed entry criteria checklist with the inspection package to the moderator. Refer to the appropriate procedure for the details of the entrance criteria checklists.

Internal Process

The moderator examines the materials to determine:

- The correct level of detail.
- Whether it is an appropriate amount to inspect.
- The availability of appropriate reference documents.

The moderator calculates the time required for the overview meeting, preparation time, and inspection meeting by dividing the size of the document to be inspected, in pages, or the code to be inspected, in lines of code, by the following estimates (these estimates are based on data related to an inspection process used in another organization and will be revised as data is collected):

- Overview. An overview should be limited to a single 2-hour meeting. If an overview longer than 2 hours is indicated by the length of the material, the moderator and the author should examine the material to divide it into separate inspection pieces. If the material indicates that an overview can be completed in a half-hour or less, the overview can be held at the beginning of the inspection meeting.

- Preparation. A`. estimate for preparation rates for preliminary design material is 12 to 15 pages per hour; for detailed design material, 8 to 12 pages per hour; and for code, 200 noncomment source statements per hour.

- Inspection. An estimate for inspection rates for preliminary design material is 10 to 12 pages per hour; for detailed design material, 8 to 10 pages per hour; and for code, 100 to 150 noncomment source statements per hour.

The moderator, project manager, or author contacts the inspection participants, tells them what is expected, and solicits available dates.

If an overview meeting is desired, it is scheduled by the moderator.

The moderator schedules the inspection using the estimated inspection meeting time computed earlier. An inspection may require multiple meetings. The length of a single inspection meeting should not exceed 4 hours in a given day. For each 4-hour meeting, it should be organized into a 2-hour meeting, a half-hour break, and then the remaining 2 hours.

The moderator or author (with approval of the moderator) distributes the inspection package consisting of the reference and inspection materials to the participants after appending the following:

- The Inspection Invitation Memo which identifies the type of inspection, time, location, time charge number, participants, and job assignments for the inspection

- Trivial Error Log

Exit Criteria

If an overview meeting is indicated, the overview meeting has been scheduled.

The inspection meeting has been scheduled.

The inspection package has been distributed to all inspection participants.

INSPECTION PLANNING RELATED PRODUCTS

Review Metrics

This is a hardcopy report summarizing the inspection results.
This product is needed before starting this activity.
This product is left unchanged by this activity.
This is a required product.

Inspection Invitation Memorandum

This memorandum is used to inform participants of an upcoming inspection.
This product is needed before ending this activity.
This product is created by this activity.

This is a required product.

INSPECTION PLANNING RELATED ROLES

Review Coordinator

Review coordinators are responsible for organizing and overseeing reviews.
The Review Coordinator Role is needed before starting this activity.
This is a required role.

Moderator

The moderator manages the overall inspection process and ensures that the requirements and intent of the inspection method are met.
The Moderator Role is needed before starting this activity.
This is a required role.

Developer

The developer is the person currently responsible for working on a product.
The Developer Role is needed before starting this activity.
This is a suggested role; the role typically participates in this activity, but you may, at your own discretion, exclude this role.

<< Remaining activities would be shown in a pattern identical to the above >>

SUMMARY OF PEER REVIEW PROCESS

It should also be noted that this method supports two types of tailoring. The first includes tailoring of guidelines and checklists to create an inspection process which adheres to the method and is relevant to the organization. You must do this type of tailoring to ensure a valid inspection process. The second type of tailoring is to modify the inspection method. This type of tailoring is optional. Additionally, tailoring of the method is subject to the limitations described below. The following discussion presents the mandatory tailoring followed by the optional tailoring.

Use of the inspection method requires that your organization establishes supporting mechanisms. These supporting mechanisms include entry criteria checklists, preparation checklists, and reinspection criteria for each artifact type that will be inspected. While there are many examples of these, you must tailor them to your organization's goals and the specific artifact being inspected.

Your organization may also tailor the classification of the defects. Typically, you classify defects, at a minimum, by type and severity. You may also classify each defect by category (i.e., missing, wrong, extra).

Because the objective of the inspection method is to find defects early in an efficient and effective manner, the focus is clearly on **defect detection**. One approach to tailoring the method, therefore, is to remove the causal analysis stage. The results of the causal analysis activity are intended to improve the development process to prevent the defect from occurring in the future. This stage emphasizes **defect prevention**. For organizations introducing the inspection process, the initial focus is probably on defect detection. The causal analysis activity could then be added in the future.

A second type of optional tailoring is to combine multiple inspections into a single inspection. This situation seems more frequent when inspecting a software artifact. An example is a change that is small in scale. The author may want to consider combining the preliminary design and detailed design inspections. Guidelines for determining when multiple inspections could be combined are:

- Combine the preliminary design and detailed design inspections when an update is made to an existing module and the preliminary design is not affected.

- Combine the detailed design and code inspections when the detailed design is done at a very detailed level and is expressed in the target language.

The moderator should be certified to run an inspection. Certification should consist of a minimum of one day of training in the inspection process and the specific role of the moderator within that process. In addition, newly trained moderators should be observed by an experienced moderator for the first 2–3 inspections. Other participants in an inspection should be given a 1-hour overview of the inspection process.

<< End of Example >>

5.7 SUMMARY

This section has shown you examples of indented lists, different graphical models, and each of the three classes of templates used at Tier 1. The above example typifies output that represents "work in progress." As the need for improved presentation increases, you can improve the way in which information is extracted from and exported by the database. For instance, an advanced script could automatically calculate and insert appropriate section, subsection, and paragraph numbers. An extremely advanced script might even imbed font size, bold face, and other font-related or formatting commands for use by a word processing package.

One of the most important characteristics of MPDM is its flexibility. Throughout this guidebook we've recommended you use an information management system for capturing and maintaining process data. A key reason is the extensive options provided by these environments. The examples in this section only hint at the almost unlimited flexibility you have in how you capture process information, and how you can export that information for use by other software packages.

Remember that guidebooks are not the only end-product you'll want to produce. Training material, for instance, is also important. By navigating the database in exactly the same order as you did for a guidebook, but only exporting the graphical fields, you will have all of the graphics from a guidebook. If you deliberately inserted pictures in the guidebook to depict key concepts, this set of pictures can be an excellent foundation for your training classes. If you want to add "Instructor Notes" to the graphics, this can be accomplished using the Transition Text fields. The point is that information management systems give you nearly unlimited options for maintaining, navigating, viewing, extending, organizing, and exporting your process data.

The next section provides further information on automated support for process definition and modeling and includes a real-world example of how a company used these techniques to produce a series of guidebooks and associated training material.

6. AUTOMATED PROCESS DEFINITION AND MODELING

This section provides additional information on automated support for process definition and modeling based on MPDM. Section 6.1 discusses the use of commercial off-the-shelf (COTS) tools that directly support MPDM techniques, Tiers 1 and 2. Section 6.2 discusses a commercially available tool that can read external files and build process models both for simulation (Tier 3) and enactment (Tier 4). Finally, Section 6.3 briefly discusses a real-world pilot project and their specific approach to automating process definition, modeling, and "push-button" generation of process guidebooks.

6.1 COMMERCIAL OFF-THE-SHELF TOOLS

The techniques described in this guidebook have been deliberately designed to be supported by relatively low cost, generic, commercial tools. Practically speaking, it is not cost-effective to pursue the definition of a complex process without automated support. In MPDM, there are advantages to be derived from each of the following classes of tools:

- Database
- Word Processing
- Presentation
- Flow Chart
- Paint

Database support is at the heart of MPDM. Process definition is an information management issue, and database software is an ideal way to manage that information.

If you want to create guidebooks, you will also need word processing software to do the final cleaning and polishing of your documentation.

Presentation software allows for easily building and maintaining "slide shows" for training classes, meetings, presentations, etc. Generic flow chart software can be especially useful for diagramming different types of process models (e.g., architectural, behavioral, interface, etc.) Paint software allows you to more easily create complex or elaborate graphical pictures.

It is important to realize that with current technologies, you can purchase a set of these tools that are capable of exchanging information. For instance, you might create a picture in a paint program and store that picture in a graphical field in the database. Similarly, you may want to import pictures onto your presentation slides. This integration of training material and guidebook material is an important principle in MPDM. By using the graphical fields in the database, you maintain the graphic representation of your process simultaneously with maintaining the textual information. This eliminates the common problem of process information evolving away from the supporting training material. Having this information coresident in a database greatly facilities parallel maintenance of pictures and text.

You need to decide when and how much to integrate your tools. At one of the spectrum are environments that support "hot-links." This feature allows you create a link, for instance, between a picture inside a paint program and that same picture embedded within a word processing document. Changing the picture inside the paint program results in an immediate and automatic update to its representation within the word processing document.

At the other end of the spectrum is manual integration. However, even at this end, the database can be used to greatly facilitate integration. For instance, there are fields within the templates that are intended to hold graphical information. If you are using a database that does not support storing or handling graphic data, then you should use these fields to name the file that holds the picture (one picture per file). Then, during export of text information from that database, whenever the script encounters a nonempty graphic field, it can export a message similar to: "<NOTE: Insert the graphic from file FILE_NAME here>." When you load the export file into the word processor, you simply search for the character string "<NOTE:" and manually import the pictures on a case by case basis (deleting the NOTE when you are done).

The issue is always one of cost-benefit. It generally takes longer to establish and maintain an integrated suite of tools, but you spend less time adding to or working on the process end-products. Conversely, *you can avoid tool integration* complexities by relying on people to conduct some or all of the final merging of, for instance, text and graphics. If you can "clean and finalize" an end-product in 1 or 2 staff days, it is likely **not** worth the effort to spend several weeks trying to improve tool-to-tool cooperation. Keep in mind that quarterly releases of a guidebook would indicate a highly aggressive update and release schedule, yet 2 staff days per quarter spent on "overhead" cleanup is a relatively small price.

You should limit your choice of automation tools to those established, mainstream products. These products are likely to be progressively improved. One advantage to this approach is that, as venders continue to find ways to improve the features of and integration between their products, you can migrate your process information to progressively more powerful and flexible environments. However, you will need to make an exception to this if the tools you need are targeted to specialized uses or represent the leading edge of a new market. An example of one such tool is discussed in Section 6.2.

6.2 SPECIALIZED TOOLS IN PROCESS MODELING: *PROCESS* WEAVER

Process support tools provide a wide range of functions and benefits for organizations seeking to improve their competitiveness and posture through improved software development practices. Process support tools can help define, manage, and provide a vehicle for improving the planning and tracking of software projects. These tools can also help monitor quality; ensure proper configuration management; and establish organization-wide, repeatable development processes.

Process support tools may also provide process enactment support which helps an organization guide the progress of software projects. Process enactment is a sequencing of process tasks, activities, and milestones defined in the process model. Process enactment is supplemented by a variety of tools and utilities that provide access to electronic mail, configuration management systems, measurement tools, metrics collection tools, and report and graph generation tools.

PROCESS **WEAVER** literature makes the following claims:

- *PROCESS* **WEAVER** can create high-fidelity models that accurately represent the generic, corporate-wide process model. These models can be effectively, efficiently, easily instantiated and customized for use on specific projects.
- It provides a rich environment that is usable by project managers, project developers, and senior management.
- It can be readily extended to include external support programs, such as measurement tools.
- Its environment capabilities together with an appropriate process model satisfy many of the key process areas for Levels 2 and 3 of the CMM.
- It provides a mechanism by which an organization may automate support for development processes and realize benefits that are in reasonable proportions to their costs. Benefits are measured by the ability to manage and improve processes in accordance with SEI's CMM. Costs can include time, money, overhead activities, etc.

From the *PROCESS* **WEAVER** information manual, this product:

> consists of a set of computer tools which allows you to model the methods and procedures in use in your company, instantiate these methodological models for specific projects, guide and support the method usage by development teams, interface with the market computer-aided software engineering (CASE) tools you are using today or intend to use in the future

The principal *PROCESS* **WEAVER** tools are:

- Agenda, the user's desktop utility from which the editors are invoked, as well as the means by which the *PROCESS* **WEAVER** activities and procedures are controlled
- The Method Editor, which is used to decompose a project into a set of activities
- The Activity Editor, which is used to parameterize an activity
- The Work Context Editor, which is used to construct objects similar to activities
- The Cooperative Procedure Editor, which is used to model the behavior of a collection of activities

Some of the process objects you can create include:

- *Methods.* These are used to model the breakdown of a process model (an activity) into smaller activities.
- *Activities.* These encapsulate the data managed by means of the process model. They are composed of inputs (e.g., documents), outputs, and roles.

- *Cooperative Procedures.* These are Petri nets and are used to model the behavior of the process model. The state transitions of a Cooperative Procedure may include a wide variety of actions, including Work Context delegations, coshell scripts, sending and waiting for external events.
- *Coshell Scripts.* The coshell is a shell interpreter written by Cap Gemini to support the special requirements of Cooperative Procedures. It is a hybrid of several languages: Lisp (for list data), C-Shell, or Pascal (control structures and syntax). Coshell scripts can be run from a command-line utility or from within Cooperative Procedures.
- *Work Contexts.* These are an instantiation of *PROCESS* **WEAVER** activities. They may be created and activated without being embedded in a Cooperative Procedure.

The following material describes a potential scenario for process modeling. Consider the simple ss model in Figure 6-1:

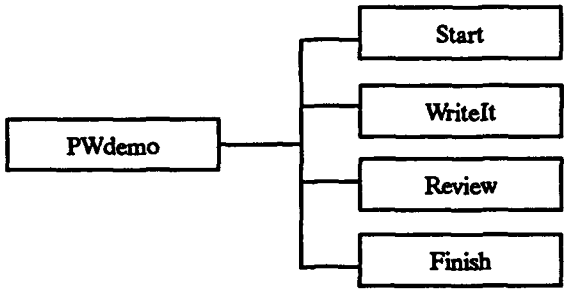


Figure 6-1. A Simplified Process Model

The activities (Start, WriteIt, Review, Finish) are performed in succession. This is easily modeled in *PROCESS* **WEAVER**:

- First, lay out the PWdemo method with the Method Editor.
- From the Method Editor, automatically generate the procedures and activities.
- For each activity, invoke the Activity Editor, supplying roles and other parameters.
- For the PWdemo activity, invoke the Cooperative Procedure Editor, to "rewire" the automatically generated procedure (which makes the dependent activities run in parallel) so that the dependent activities run in succession.

The method can then be instantiated (i.e., run) from the user's Agenda.

Development does not proceed in a straight line except in the most ideal environment. The simple model lacks the following essential characteristics:

- Provision for more than one person for each role
- Handling rejection, revision, and other iterating processes
- Interfacing with a wider environment than the *PROCESS* **WEAVER** Agenda

You can refine the model by adding more features to address these areas.

1. The simple model has only three roles, with one person per role: Manager, Author and Reviewer. Actual processes tend to be more complex. For instance, you may want to model three or more reviewers, and add a technical editor to check documents. To do this:

   a. Modify the project startup activity to separate the roles that initiate a project from those that allocate manpower. *PROCESS* **WEAVER** assigns roles to variables that are specified at process instantiation time. They are stored as coshell variables. You can modify the method so that the roles were no longer parameters of the method, but are instead set via a coshell script which read the values via an activity's data-entry panel.

   b. Allow the personnel specified for the method's roles to be lists of *PROCESS* **WEAVER** users. Because the coshell automatically handles lists, the main effect of this change is to the Cooperative Procedure for the Review activity. The revised model will distribute the review to each listed Reviewer and collect their responses.

   c. Add a TechEdit (technical editing) activity to the model, with an associated Editor role.

2. To model iteration and branching, use the facility of *PROCESS* **WEAVER** that allows multiple types of responses in the Work Context and Activity data-entry forms. A user responds to an Activity by clicking on a push-button in the data-entry form. Each button (there may be any number) has an identifier that can be tested by a coshell script. To keep the model manageable, split the Cooperative Procedure into a hierarchy. The top level of the resulting model is illustrated in Figure 6-2.

   NOTE: Because this version of *PROCESS* **WEAVER** does not have facilities for printing, this diagram was drawn using Interleaf.

   a. Error checking can be performed in INITIALIZE and TS_Start. If an error is discovered in the Startup activity, the model branches to STOP.

   b. The Review activity (invoked in TS_Review) collects the responses of the reviewers. All Reviewers must agree that the document needs no further change; otherwise it is sent back to the Author for revision.

   c. There are three possible branches in the TS_TechEdit activity: pass (no change required), minor revisions required, rewrite required. The minor-revision loop is embedded in the TechEdit activity's Cooperative Procedure.

   d. There are two possible branches in the TS_Finish activity: ship and no-ship, resulting in the states PE_Finish and PE_Giveup, respectively.

3. One characteristic of *PROCESS* **WEAVER** is that it performs notification only within its own environment. Users must have the Agenda program running (even if in icon mode) to receive notification. Computer users are more accustomed to receiving notification via electronic mail. You can augment your model to send electronic mail in addition to the automatic notification performed by the Agenda program.

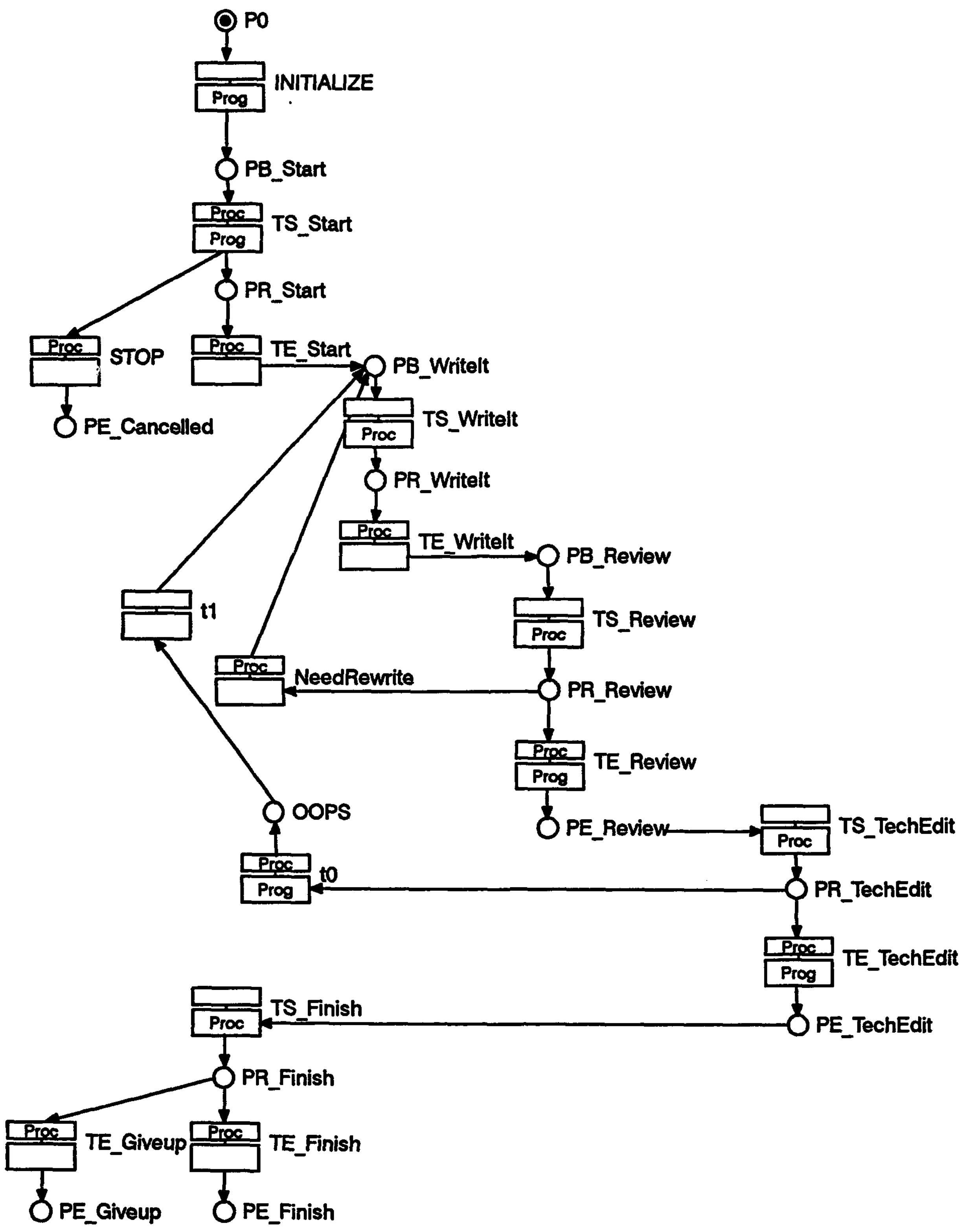


Figure 6-2. Refined Cooperative Procedure Model — Top Level

a. For each activity, electronic mail is sent to the users who will be notified by the Agenda program (e.g., the Author in the WriteIt activity, Reviewer in the Review activity).

b. Reviewer comments are mailed to the Author. This is actually simpler than trying to maintain a single collection of review comments shared by all reviewers.

c. Electronic mail notifies manager on completion of the review (in TE_Review).

You may want to simulate the model at several stages—each time adding functionality. The simulations can highlight errors in the model, which can then be corrected.

It should be noted that *PROCESS* **WEAVER** includes programs and scripts that enable it to work in conjunction with Microsoft Project (a scheduling tool).

Clearly, *PROCESS* **WEAVER** can be considered a state-of-the-art tool. This class of tools can be quite valuable for those conducting analytical process research and for those who have highly integrated, technically advanced environments.

If you are seriously considering this type of environment, you will want to select a database that supports managing and exporting process information in a manner that allows for import by your process simulation or enactment tool(s).

6.3 VITRO PILOT PROJECT EXPERIENCE

During 1993, Vitro Corporation used the *Process Definition and Modeling Guidebook* (Software Productivity Consortium 1992) as a basis for their process definition. Numerous insights were gained from that pilot project, and the templates and techniques have been greatly extended as a result of that experience. Sections 6.3.1 through 6.3.5 describe how Vitro automated their process definition work, including example template screens and guidebook material.

6.3.1 Background and Overview

Building on many years of effort in software quality, process improvement, and a corporate-wide commitment to TQM, Vitro Corporation established a new Software Center of Excellence in 1992 as a means of consolidating Vitro's software resources and deploying them more effectively in pursuit of major software business opportunities.

One of the first tasks of this new Center of Excellence has been to define and document Vitro's software engineering process. The SEPG within the Center of Excellence has taken on this task. SEPG Team Leader, Walt Greenspon, and other team members are using the Consortium's *Process Definition and Modeling Guidebook* and the expertise of Consortium staff to define a comprehensive software engineering process to be used initially by the 200 software engineers at Vitro's Silver Spring, Maryland headquarters, and subsequently to be used throughout Vitro's field offices.

6.3.2 Process Maturity Equals Competitiveness

With approximately 5,500 employees and over $400 million in annual sales, Vitro enjoys a reputation as a leading provider of engineering systems and services, primarily to government customers. Ongoing process improvement and TQM efforts have long been seen by Vitro executives as critical activities in the continued expansion of the company and further diversification of its markets.

Although Vitro's TQM efforts predate the CMM of the SEI, Mr. Greenspon and other software managers saw value in the CMM as an initial framework for a software-specific process improvement plan, one much in keeping with Vitro's corporate-wide TQM philosophy.

Following an SEI Software Process Assessment (led by Mr. Greenspon and assisted by the Consortium) in 1992, the SEPG has been coordinating action plan implementation, which includes the development of a series of guidebooks rigorously defining and documenting Vitro's software engineering process, with specific mappings to the Key Process Activities of the CMM. Once this family of guidebooks is developed and implemented, Vitro plans a follow-on assessment in 1994. The Consortium's *Process Definition and Modeling Guidebook* is serving as the basis for this series of Vitro-specific guidebooks.

Noting the growing emphasis in government RFP's on demonstrated process maturity and capability, Mr. Greenspon sees this process definition activity as further aligning Vitro with good engineering practices and increased business opportunities.

Vitro is also mapping appropriate parts of their process specifically to the SEI CMM. They have carefully allocated CMM requirements to their process definitions in a way that supports "two-way vision." This allows Vitro to examine their process to see what CMM requirements have been satisfied, or look at the CMM and see what parts of the Vitro process address it. This not only improves their software capabilities, but also helps Vitro demonstrate those capabilities to their customers.

6.3.3 Family of Guidebooks

Vitro's SEPG has laid out a flexible, extensible process documentation architecture. Process documentation for each broad area, such as Software Development, Independent Verification and Validation, and Reengineering, will be packaged into boxed sets. Individual guidebooks within each set focus on specific topics, including Product Engineering, Project Planning, Project Management, Configuration Management, and Quality Assurance. Other, separate guidebooks are planned in areas such as training, SEPG activities, and policies and procedures. A Process Architecture Description guidebook serves as an overall guide, or roadmap, to the entire guidebook family.

Mr. Greenspon and other SEPG team members have automated portions of the guidebook development and production process through the use of a relational database and windows environment. These tools are being used to manage process information in the guidebooks and to navigate through a series of textual "templates" for process definition and description, as outlined in the Consortium's guidebook.

Vitro plans a series of guidebooks that iteratively expand in both scope and detail. The approach allows Vitro to capture lessons learned and insights into process improvements and update their process approach and documentation accordingly.

6.3.4 For Software *and* System Engineers

The SEPG envisions the eventual user base of their guidebook series to extend beyond Vitro's software engineers. Mr. Greenspon and other SEPG team members are looking to move this process into other projects. Initial work is already underway in extending this process to define a process for systems engineering. One key objective is the ability to have all participants in systems and software engineering cooperating and working more closely than ever before. Vitro's SEPG collaborates with the system engineers to help ensure that the process definitions they use fit into the overall process architecture model.

6.3.5 Examples

Although Vitro wanted to manage their process information in an information management repository, they also wanted process documentation that looked and read like guidebooks—not like a dump of database fields. MPDM explicitly addresses this problem through the flexibility of database environments and through the transition text fields. Consequently, the output of the process database can have an appearance much closer to what people have come to expect in quality process guidebooks.

Note in the examples—provided by Vitro—of a table of contents and three pages from a process guidebook, that the appearance largely conceals the fact that the information was automatically exported from a database, and the word processor was only used for final clean-up and merging of graphics. Vitro has found that in general, it takes 2 to 4 hours of final clean-up and merging for every 100 pages of guidebook. Examples 6-1 through 6-7 include copyrighted material and are reprinted by permission of Vitro Corporation.

To support information management, Vitro selected Paradox, by Borland International. One of the strengths of the Paradox software is its support for creating custom, graphical, user interfaces for data entry and management. Example 6-5 hows the "main menu" screen that Vitro's process engineers use to access and update the various classes of templates.

Examples 6-6 and 6-7 show an event (or activity) template maintenance screen and a role template screen. There are certain desirable software support features for designing and using these types of screens. For instance, it is highly desirable for text fields to have a default size and also, if you "click open" that field, a full screen size that facilitates entering and editing text. Also desirable, as shown in figure <role screen> is the ability to have your own "pull-down" menus associated with different parts of the template.

Additionally, be sure to consider the advantages of automated environments that have strong "cut and paste" support. MPDM advocates a cyclic approach to adding progressively more detail to your process model. During a cycle, it is typical to turn selected atomic templates (i.e., templates without children) into parents and create children under those parents for capturing additional detail. Some of that detail is often already on the parent template, and powerful cut and paste facilities can greatly facilitate redistributing that information to the appropriate child templates.

Copyright © 1993 by Vitro Corporation. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the Vitro Corporation. Images copyright, New Vision Technologies Inc.

CONTENTS

Foreword i

1. Managing the Project Software Configuration 1

2. Identifying the Configuration 4

3. Controlling the Configuration 6

    3.1. Administering a Configuration Management Library System 7

    3.2. Performing Informal Configuration Control 9

        Establishing Informal Control of Configuration Items 10

        Incorporating Changes to Informally Controlled Configuration Items 11

    3.3. Performing Formal Configuration Control 12

        Establishing Formal Control 14

        Managing Changes to Formally Controlled Configuration Items and Baselines 15

            Processing Problem Reports and Change Requests 16

        Controlling the Release of Products 18

4. Performing Configuration Status Accounting 20

5. Auditing the Configuration 22

A. Product References A-1

B. Role References B-1

Example 6-1. Table of Contents

1. Managing the Project Software Configuration

This section introduces the project software Configuration Management (CM) discipline and answers the questions; "What is CM?", "Why do CM?", and "Who does CM?". Four major CM processes are introduced, along with a statement of the goals of CM performance.

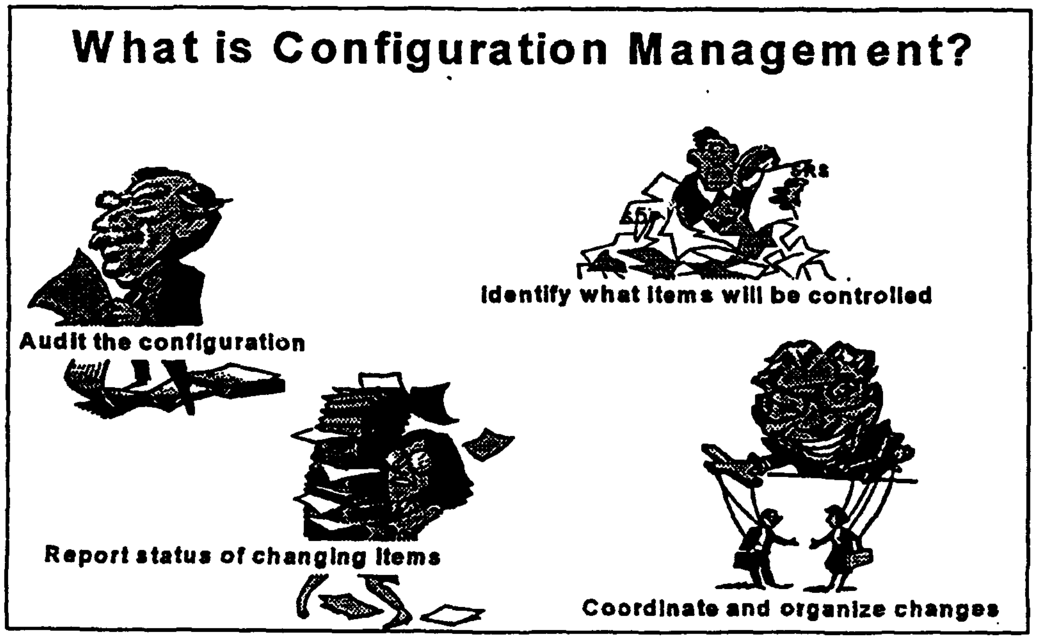
Entry Criteria:
This activity begins upon approval of the project's Software Configuration Management Plan.

Internal Processing:
Software Configuration Management (CM) involves identifying the configuration of the product (software and associated documentation), systematically controlling changes to that configuration, and maintaining its integrity and traceability throughout the software life cycle. The documented and approved project Software Configuration Management Plan, created during project planning, is used as the basis for performing the CM activities.

During performance of the CM process, certain measurements are made and used to determine the status of the CM activities, such as:

- Completion of milestones for the CM activities compared to the plan
- Work completed in the CM activities
- Effort expended in the CM activities
- Funds expended in the CM activities

These measures are summarized in the Configuration Management Status Report, prepared and distributed by the CM staff.


What is Configuration Management?
Identify what items will be controlled
Audit the configuration
Report status of changing items
Coordinate and organize changes

Example 6-2. Process Guidebook Text, Page 1

5. Auditing the Configuration

This section describes the fourth of four major CM processes, Configuration Audits. Included is a discussion of why configuration audits are performed, how configuration audits are accomplished, and what type of configuration audit documentation should be created.

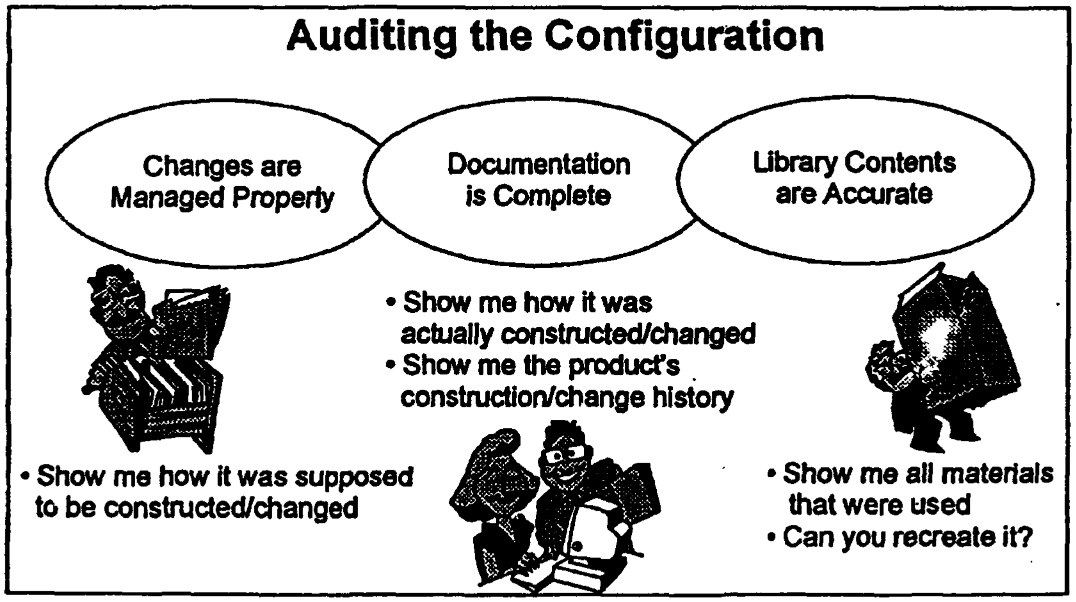
Entry Criteria:
According to Configuration Audit schedules contained in the project Software Configuration Management Plan.

Internal Processing:
The Project Software Configuration Management Staff periodically performs software configuration audits to ensure that the CM practices and procedures are rigorously followed and to ensure the integrity of the software configuration items and baselines. This audit function is an integral part of the CM process. The audit team should consist of CM personnel that are not directly involved in the tasks being audited. The audit verifies that:

- changes to each configuration item and baseline have been properly managed
- the change documentation is complete
- the contents of the CM library are accurate.

To verify the above, the auditors will, for example:

- compare Problem Report & Change Request contents and status to CCB Minutes, Configuration Status Accounting Reports, and actual CM Library contents
- compare CM Library labeling to the Software Project Configuration Index
- compare actual CM Library contents to the Library Contents List, the Library Check Out/In Log, Informal Control Submittal/Update forms, Problem Reports, Change Requests, and CCB Directives.


Auditing the Configuration
Changes are Managed Properly
Documentation is Complete
Library Contents are Accurate
• Show me how it was actually constructed/changed
• Show me the product's construction/change history
• Show me how it was supposed to be constructed/changed
• Show me all materials that were used
• Can you recreate it?

Example 6-3. Process Guidebook Text, Page 2

For each planned configuration audit, the following should be defined:

- the objective
- the configuration items and baselines under audit
- the schedule of the audit
- the procedures for conducting the audit
- the audit participants
- documentation required to be available for review or to support the audit.

For each audit, the audit team prepares and distributes a Configuration Audit Agenda, records audit data and discrepancy information, and prepares and distributes a Configuration Audit Report.

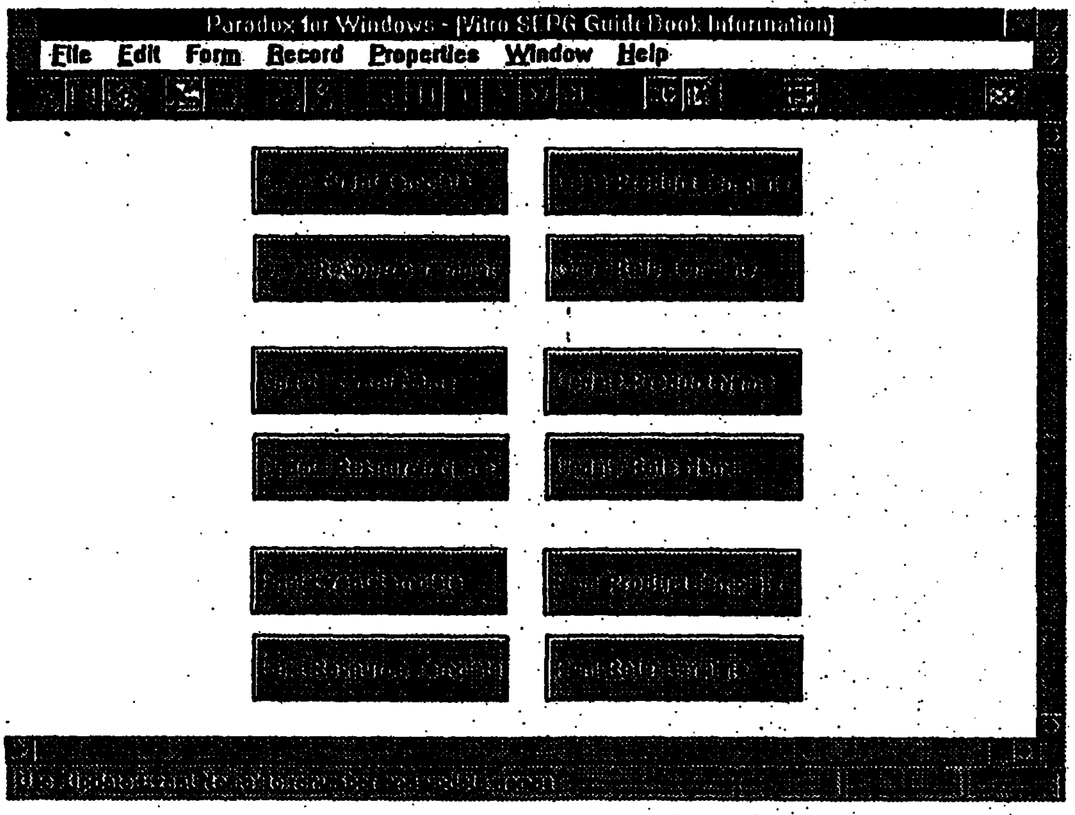
A successful audit must be performed prior to key checkpoints in each development phase (Management Review, as defined in the Product Engineering activities) and before every major baseline change.

Exit Criteria:
Audit results have been reported.

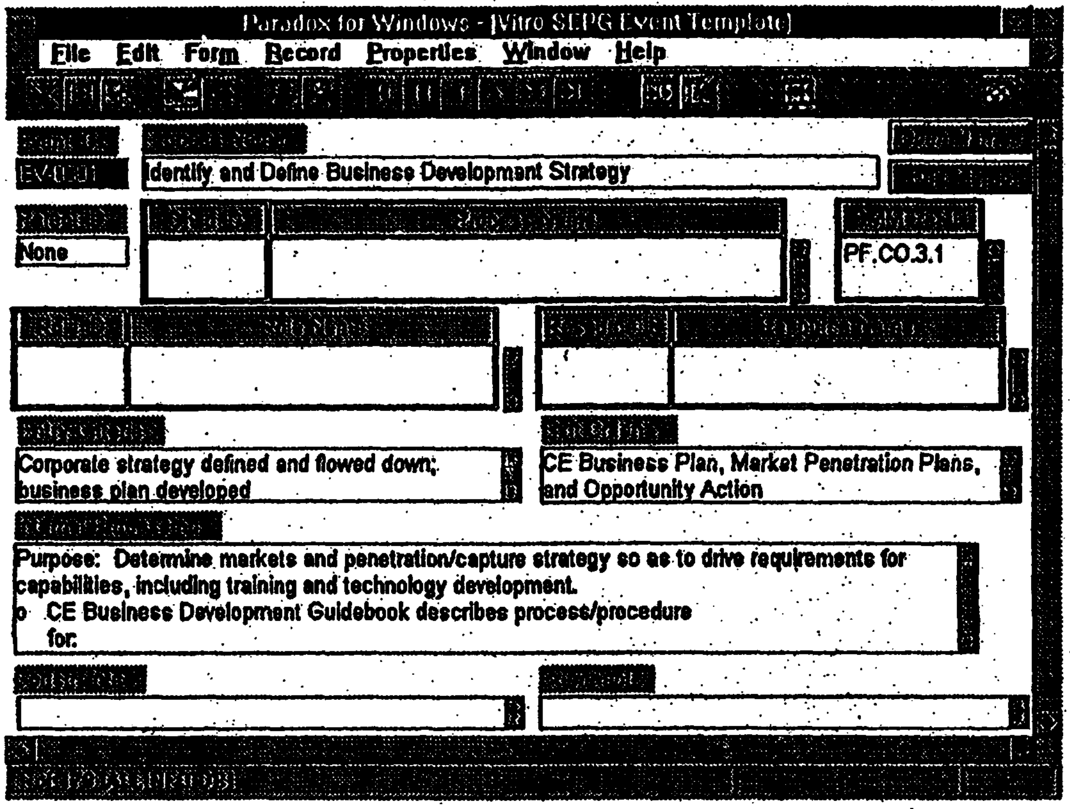
Product References:	Usage:
Configuration Audit Agenda	Created
Configuration Audit Report	Created
Software Project Configuration Index	Used as Resource
Informal Control Submittal/Update	Used as Resource
Problem Report/Change Request	Used as Resource
Library Contents List	Used as Resource
Library Check Out/In Log	Used as Resource
CCB Minutes	Used as Resource
CCB Directive	Used as Resource
Configuration Status Accounting Reports	Used as Resource

Role References:
Project Configuration Management Staff

Example 6-4. Process Guidebook Text, Page 3


Paradox for Windows - [Vitro SEPG GuideBook Information]
File Edit Form Record Properties Window Help

Example 6-5. Process Database Main Menu


Paradox for Windows - [Vitro SEPG Event Template]
File Edit Form Record Properties Window Help

EV-[illegible]

Identify and Define Business Development Strategy

None

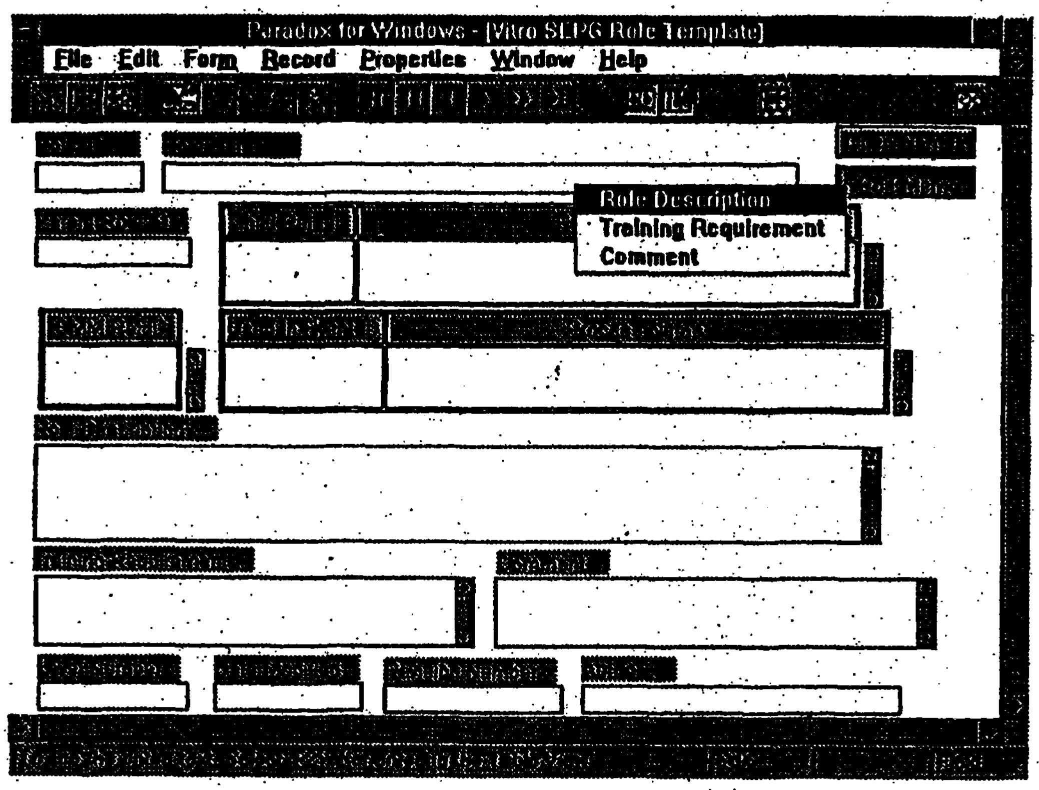
PF.CO.3.1

Corporate strategy defined and flowed down; business plan developed

CE Business Plan, Market Penetration Plans, and Opportunity Action

Purpose: Determine markets and penetration/capture strategy so as to drive requirements for capabilities, including training and technology development.
o CE Business Development Guidebook describes process/procedure
for:

Example 6-6. Process Database Event (Activity) Screen


Paradox for Windows - [Vitro SEPG Role Template]
File Edit Form Record Properties Window Help
Role Description
Training Requirement
Comment

Example 6-7. Process Database Role Screen

6.4 SUMMARY

This first volume of this guidebook has presented you with all the fundamental principles and techniques of MPDM. You have sufficient information to begin defining and modeling your process in an manner that directly supports the automated creation, maintenance, and generation of process-related guidebooks, training materials, etc.

There is, of course, the option for much more complexity and formality. Volume 2 of this guidebook provides further information on how to approach Organizational Process Definition and presents considerably more material on additional templates and fields.

It should be stressed that Volume 1 of the guidebook assumes you understand the organizational and human factors that surround and critically affect the success of any effort at process definition (or process improvement in general). Volume 2 of this guidebook does not make this assumption, and considerable material is provided in Volume 2 on how process definition occurs within the much larger contexts of process engineering, process improvement, and organizational change.

This page intentionally left blank.

LIST OF ABBREVIATIONS AND ACRONYMS

CASE	computer-aided software engineering
CM	configuration management
CMM	Capability Maturity Model
DOD	Department of Defense
ESP	Evolutionary Spiral Process
ETVX	Entry-Task-Validation-eXit
FAA	Federal Aviation Administration
IDEF	Integrated DEFinition
IE	Improvement Efforts
ISO	International Organization of Standardization
MPDM	Managed Process Definition Methodology
NASA	National Aeronautics and Space Administration
NSA	National Security Agency
OPD	Organizational Process Development
PAD	Project Application Development
PASTA	Process and Artifact State Transition Abstraction
PAT	Process Action Team
PERT	Program Evaluation and Review Technique
PLD	Product-Line-Based Product and Process Development
PPA	Program Process Architecture
QA	quality assurance
R & D	research and development

RFP	Request for Proposal
RIN	Role Interaction Nets
SADT	Structured Analysis and Design Technique
SDT	State Transition Diagram
SEI	Software Engineering Institute
SEPG	Software Engineering Process Group
SPICE	Software Process Improvement Capability dEtermination
TQM	Total Quality Management
V & V	verification and validation
VCOE	Virginia Center of Excellence for Software Reuse and Technology Transfer

GLOSSARY

Class template	Class templates are derived from meta-class templates and add information unique to that template.
Constraint	Process constraints describe the limiting conditions associated with the activation, performance, or cessation of an event. Whereas supports can be viewed as those things required to enable or make the right things happen, constraints can be viewed as those things required to disable or prevent the wrong things from happening. In this guidebook, constraints have been divided into two general types: internal constraints and external constraints.
Contributing template	These are any non-leaf-node templates.
Cycle	A traversal of all five sectors of the spiral model, which donates that some aspect of the product has matured by a specific amount.
End-product	The product that results from a process definition and modeling effort. Examples include guidebooks, training material, subsections of proposals, operations manuals, project plans, etc.
Estimate of the situation (EoS)	A document that identifies the project's goals, strategies, product and process assumptions, and the assets available for performing a project.
External constraints	External constraints include all factors that may limit or constrain how an activity proceeds, that are not directly attributable to local authority (which are modeled as internal constraints). Examples of external influences that may constrain an activity include quality requirements, corporate standards, division policies, engineering procedures, process guidelines, and management directives. External constraints differ from internal constraints in that they are typically not subject to discretionary use—they are intended to be, and expected to be, explicitly followed, regardless of project-specific issues.

Final template	These are the leaf-node templates.
Foundation template	All templates are derived from a common foundation template.
Guidance	The use of a process definition (constraint) by an observer or process agent to provide the enacting process agent with the legal set of process step options at any point of the enactment of the observed process. This may involve process cues, process interaction, or process management.
Internal constraints	Internal process constraints are typically managerial in nature and usually take the form of authority and permission. Examples of internal constraints include management authority or permission required before an event can commence. Internal constraints also convey authority to roles to suspend events, cancel activities, recommence activities, cease activity, etc. In all cases, internal constraints are always coupled with a role (typically a role signifying lead or managerial responsibility, but in all cases a role signifying—by definition—some form of authority). As a rule, internal constraints are those constraints that you have authority to change, countermand, enforce, etc.
Meta-class template	Meta-class templates inherit common fields from the foundation template.
Policy	A guiding principle; a process constraint, usually at a high level, that focuses on certain aspects of a process and influences the enactment of that process.
Precision	The degree to which the process definition completely specifies all the actions needed to produce accurate results. That is, a precisely defined process, executed with fidelity, produces an accurate result.
Predictability	An indication that either the process is intended to terminate and does terminate or that the process is intended to be nonstop and that it does continue until terminated by a control process (or its agent).
Process	A series of actions or operations conducing to an end. A series of actions intended to reach a goal, possibly resulting in products.

Process architecture	A conceptual framework for incorporating process elements in consistent ways (or for signaling that the process element is incompatible with the architecture). A framework within which project-specific processes are defined.
Process control	The external influence over process enactment by other enacting processes. This influence may be driven by process evaluation and may be through control of the process enactment state, reassignment of resources, or change of process goals through process evolution.
Process definition	An instantiation of a process design for a specific project team or individual. It consists of a partially ordered set of process steps that is enactable. Each process step may be further refined into more detailed process steps. A process definition may consist of (sub)process definitions that can be concurrently enacted. Process definitions, when enactable by humans, are referred to as process scripts. Process definitions for nonhuman enactment are referred to as process programs.
Process evolution	The evolution of process definitions (static) as well as the evolution of enacting processes (dynamic), e.g., nonstop processes. Both static and dynamic change must be managed to ensure stability of the process and control over the process results.
Process model	A possibly partial process definition for the purpose of modeling certain characteristics of an actual process. Process models can be analyzed, validated, and, if enactable, simulates the modeled process. Process models may model process architecture, design, project plans, etc. Process models are at times used to predict process behavior.
Process modeling	Process modeling both extends and constrains process definition by requiring that the process model adheres to a predefined set of objects, relationships, methods, and structural conventions, the latter of which are often rendered graphically.
Process representation	A general term referring to the combined or sequential efforts of jointly performing process definition and process modeling.

Process supports	A process support is any non-throughput item that is needed by an activity for the activity to be performed. Activities need products, as that typically is the purpose of activities: to accept one or more products, modify, manipulate, inspect, and possibly create one or more new products, and pass those along to other activities. However, more is needed by an activity than just the products. These nonproduct items are all modeled as supports. Two common types of support include roles and resources.
Products	Products represent the vast majority of artifacts that pass through a process. Examples include code modules, end-user guidebooks, circuit boards, and anything else tangibly produced by a process. Products can be decomposed into subproducts, sub-subproducts, etc.
Project	An enactable or enacting process whose architecture has control processes (project management) and enacting processes performing the project tasks.
Project management	An enactable or enacting process whose goal is to create project plans and, when authorized, instantiate them, monitor them, and control their enactment. These responsibilities are commonly known as project planning (i.e., development of process plans) and project control (i.e., process evaluation of plan information and process control to make adjustments, if necessary).
Project manager	A human agent enacting the control process responsible for the execution of a project.
Redundancy	A process task or step that is not required by an error-free enactment. Redundancy thus compensates for human or other errors in process enactment.

Research	Research is a by-product of a process; but it differs from products in that research is considered intangible. If for instance, the research leads to a technical paper, that technical paper is modeled as a product. However, if experiments or investigations are being performed within one or more events, but nothing tangible is available as evidence of the work, the throughput can still be explicitly modeled as a research (intangible) artifact. As with products, research can be decomposed into subresearch, subsubresearch, etc. This decomposition is captured within a model by the inclusion relation.
Resources	Resources are nonhuman items needed to support an event. Examples include equipment, office space, supplies, funding, etc. All items that might be required to support an event can be modeled as resources. Resources can be decomposed (using the inclusion relation) so that while one level of event abstraction shows that the training building is required, at a lower or more detailed level of abstraction the support might show that only a small classroom is actually required.
Robustness	The degree to which the process rejects unauthorized process control and/or modification (intrusion).
Role	Roles commonly represent either individual humans or humans working in concert toward a common goal or set or goals. Consequently, "programmer," "manager," "clerk," etc., all define roles that can be assumed by individuals. However, "programming team," "inspection department," and "quality assurance division" also define roles. In the latter case, the roles are essentially organizational roles as opposed to individual. For process definition, roles can be defined at all levels of abstractions.
Spiral	One or more cycles.
Subclass template	Subclass templates are used to further refine and distinguish process information.
Task	A process (step), typically enacted by a human, requiring process planning and control.

Tiers of Usage	Tiers of usage separate process definition work by degree of formality. Tier 1 usage requires the least degree of formality. Tier 4 usage requires a high degree of formality.

REFERENCES

1986
A Spiral Model of Software Development and Enhancement. *ACM Software Engineering Notes* 11:22–42.

1988
A Spiral Model of Software Development and Enhancement. *IEEE Computer* 21:61–72.

Boehm, B., and F. Belz
1989
'Experiences With the Spiral Model as a Process Model Generator," In *Proceedings of the 5th International Software Process Workshop*, pp. 43–45.

1992
"CASE & the Management of Risk." Presented at the CASE World Conference, Santa Monica, California, 18-20 February.

Coleman, Glenn L., Charles P. Ellison, Gentry P. Gardner, Daniel L. Sandini, and John W. Brackett
1990
Experience in Modeling a Concurrent Software System Using STATEMATE. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering, pp. 104–108.

Department of Defense
1985
Technical Reviews and Audits for Systems, Requirements, and Computer Programs, DOD-STD-1521B. Washington, D.C.: Department of Defense.

1988
Military Standard: Defense System Software Development, DOD-STD-2167A. Washington, D.C.: Department of Defense.

Fagan, M.E.
1986
"Advances in Software Inspections." In *IEEE Transactions on Software Engineering*, Volume SE-12, Number 7, pp. 744–751.

Feiler, Peter H., and Watts S. Humphrey
1992
Software Process Development and Enactment: Concepts and Definitions, CMU/SEI-02-TR-4. Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University.

Grove, Andrew S.
1983
High Output Management. New York, New York: Random House.

Harel, David
1988
Statecharts: A Visual Formalism for Complex Systems. Department of Applied Mathematics, The Weizmann Institute of Science, Rehovat, Israel. 1986. (Eventually published under the same title in 1987 in *Science of Computer Programming,* 8(1987):231–274.)

Henderson, W., and P. Taylor 1991	Embedded Processes in Stochastic Petri Nets. *IEEE Transactions on Software Engineering* 17, 2.
Implementation Management Associates 1992	Accelerating Change Workshop. Brighton, Colorado: IMA, Inc.
Kellner, M. 1989	Representation Formalisms for Software Process Modeling. *Proc. 4th International Software Process Workshop*.
Kolman, Bernard, and Robert C. Busby 1984	*Discrete Mathematical Structures for Computer Science.* Englewood Cliffs, New Jersey: Prentice-Hall Inc.
Levis, Alexander 1992	Class lecture notes during ECE590/SYST659 Spring 1992 course at George Mason University.
Marca, David A., and Clement L. McGowan 1988	*SADT: Structured Analysis Design Technique.* McGraw-Hill Book Company.
Paulk, Mark, William Curtis, Mary Beth Chrissis, and Charles V. Weber 1993	*Capability Maturity Model for Software, version 1.1* CMU/SEI-93-TR-24. Pittsburgh, Pennsylvania: Software Engineering Institute.
Radice, Ronald A., and Richard W. Phillips 1988	*Software Engineering: An Industrial Approach.* Englewood Cliffs, New Jersey: Prentice-Hall Inc.
Redwine, S.T. 1991	"Process Architecture Issues," In *Proceedings of the Seventh International Software Process Workshop*, Yountville, California, pp. 117–120.
Redwine, S.T. and W.E. Riddle 1985	"Software Technology Maturation." In *Proceedings 8th International Conference on Software Engineering,* IEEE.
Sage, Andrew 1993	"Systems Engineering for Software Intensive Systems." Presentation by Software Productivity Consortium, September 23.
Sanden, Bo 1992a	*Software Systems Construction: Sequential and Concurrent Designs Implemented in Ada.* Pre-Published Textbook used at George Mason University for Spring 1992. Material is the pıoperty of Bo Sanden and Prentice Hall.
1992b	*3.0 Statecharts*. Supplemental Handout #2 for INFT821. Spring 1992: George Mason University.

Singh, Baldev 1992	*Interaction Roles: A Coordination Model*, CT-084-92. MCC Technical Report.
Singh, Baldev, and Gail L. Rein 1992	*Role Interaction Nets (RINs): A Process Description Formalism*, CT-083-92. MCC Technical Report.
Software Productivity Consortium 1992	*Process Definition and Modeling Guidebook*, SPC-92041-CMC, version 01.00.02. Herndon, Virginia: Software Productivity Consortium.
1993a	*Managing Process Improvement: A Guidebook for Implementing Change*, SPC-93105-CMC, version 01.00.06. Herndon, Virginia: Software Productivity Consortium.
1993b	*Using New Technologies: A Technology Transfer Guidebook*, SPC-92046-CMC, version 02.00.08. Herndon, Virginia: Software Productivity Consortium.
1994	*Process Engineering With the Evolutionary Spiral Process Model*, SPC-93089-CMC, version 01.00.06. Herndon, Virginia: Software Productivity Consortium.
Venkatraman, N. 1991	"IT-Induced Business Reconfiguration." *The Corporation for the 1990's.* Edited by Michael S. Scott Morton. New York: Oxford University Press, pp. 122-58.

This page intentionally left blank.

BIBLIOGRAPHY

Britton, K.H., R.A. Parker, and D.L. Parnas. "A Procedure for Designing Abstract Interfaces for Device Interface Modules," In *Proceedings, 5ICSE*. pp. 195–204, 1981.

Clements, P.C., R.A. Parker, D.L. Parnas, J.E. Shore, and K.H. Britton. *A Standard Organization for Specifying Abstract Interfaces.* NRL Report 8815, June 14, 1984.

Curtis, B. *Modeling, Measuring, & Managing Software Development Process. The M3 Life Boat for Software Tarpits.* Tutorial #4, The 13th Internal Conference on Software Engineering, 1991.

Feiler, Peter, and Watts Humphrey. *Software Process Definitions Draft Document.* Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie-Mellon University, 1991.

Guindon, R. *A Framework for Building Software Development Environments: System Design as Ill-structured Problems and as an Opportunistic Process.* MCC Technical Report STP-298-88, 1988.

Kirby, J. Jr., R.C.T. Lai, and D.M. Weiss. "A Formalization of a Design Process." In *Proceedings, 1990 Pacific Northwest Software Quality Conference.* Oct. 29–31, 1990:93–114.

Osterweil, L. "Software Processes Are Software Too " In *Proceedings, 9ICSE.* March 1987.

Parnas, D.L. "On the Criteria to be Used in Deco [illegible] ıg a System into Modules." *Communications of the ACM* 15,12 (1972):1053–1058.

Parnas, D.L., and P.C. Clements. "A Rational Design Process: How and Why to Fake It." *IEEE Transactions on Software Engineering* (February 1986).

Potts, C. "A Generic Model for Representing Design Methods." In *Proceedings, 11ICSE*, 1989.

Potts, C., and G. Bruns. "Recording the Reasons for Design Decisions." In *Proceedings, 10ICSE*, April 1988.

Rendes, Barry, and Ralph M. Stair, Jr. *Quantitative Analysis for Management.* Third Edition. Allyn and Bacon Inc., 1988.

This page intentionally left blank.